MASTER EN SCIENCES INFORMATIQUES

La théorie Vision-compréhension appliquée aux patrons de conception approche expérimentale

Van den Plas, Bertrand

Award date:
2009

Awarding institution:
Universite de Namur

Facultés Universitaires Notre-Dame de la Paix de Namur
Faculté d'Informatique

La théorie « Vision-Compréhension » appliquée aux patrons de conception.

Approche expérimentale

Bertrand Van den Plas

Mémoire présenté en vue de l'obtention du grade de Maître en Informatique
Année académique 2008-2009

Résumé

Le génie logiciel existe depuis de nombreuses années. Cette discipline de l'informatique, qui est également une science selon [HALS08], étudie notamment l'activité de compréhension logicielle nécessaire à n'importe quelle tâche de maintenance d'un logiciel. Cependant, rares sont les théories expliquant des faits bien connus de cette activité et permettant d'en prédire de nouvelles.

La théorie « Vision-Compréhension », récemment proposée par Guéhéneuc [Gué06], suggère une explication à l'acquisition des données par la vue lors de l'activité de compréhension logicielle. L'auteur en propose une application pour expliquer la rapide adoption des patrons de conception par la communauté du génie logiciel. Cette adoption serait due à l'aide que les patrons de conception fournissent au programmeur engagé dans une activité de compréhension logicielle.

Au travers d'une expérience où nous utilisons un oculomètre, nous avons tenté d'apporter un élément permettant de corroborer cette qualité supposée des patrons de conception, et par la même occasion, l'application de la théorie « Vision-Compréhension » aux patrons de conception et, enfin, la théorie elle-même.

Ce mémoire tente de mieux cerner chacun des domaines, et ils sont nombreux, touchant à la réalisation d'une expérience en génie logiciel avec utilisation d'un oculomètre et de notations graphiques telles qu'UML. Ensuite, un compte-rendu de l'expérience que nous avons réalisée est proposé.

Mots clés : génie logiciel empirique, compréhension logicielle, maintenance de logiciel, oculométrie, diagramme de classes UML, patron de conception, acquisition d'information.

Abstract

Software engineering exists for many years. This discipline of computer science, which is also a science according to [HALS08], study the necessary programme comprehension activity for any software maintenance. However, few theories are capable of explaining well known facts during this activity and are capable of predicting new ones.

The Vision-Comprehension theory, recently introduced by Guéhéneuc [Gué06], explains the acquisition of information through sight during the program comprehension activity. The author proposes an application of his theory to explain the rapid adoption of design patterns by the software engineering community. This adoption would be a consequence of the fact that design patterns provide an aid to the software engineer engaged in an program comprehension activity.

Through an experiment using an eye-tracking system, we attempt to provide evidence to support this alleged quality of design patterns and, in the same time, the application of the theory for design pattern, and finally, the theory itself.

This master thesis attempts to better define the numerous fields affecting the realization of an experiment in software engineering with an eye-tracking system and graphic notations as UML. Then, an account of the realized experiment is proposed.

Keywords : empirical software engineering, program comprehension, software maintenance, eye-tracking, UML class diagram, design pattern, acquisition of information.

Je tiens à remercier tous ceux qui, de près ou de loin, m'ont permis de réaliser ce travail et en particulier les participants à mon expérience.

Je remercie plus particulièrement mon promoteur Naji Habra pour ses conseils et la confiance qu'il m'a accordée.

Je tiens également à remercier mon maître de stage Yann Gaël Guéhéneuc pour son accueil à l'École Polytechnique de Montréal, pour son suivi et ses conseils précieux, tant durant mon stage que pour la rédaction de ce mémoire.

Je remercie également André Brissette pour nous avoir ouvert les portes de Pyxis et Gérardo Cepeda pour son aide durant l'expérience.

Enfin, je remercie ma compagne Catherine pour sa patience et son soutien ainsi que ma maman, infatigable dans ses relectures.

À tous, merci.

Table des matières

Liste des tableaux

Table des figures

Liste des acronymes

Anova	Analyse de la variance.
DS	Densité spatiale.
Erreur α	Erreur de type I.
GoF	Gang of Four.
H_0	Hypothèse nulle.
H_1 ou H_α	Hypothèse alternative.
MT	Matrice de transition.
RFO	Rapport de fixation (on target all).
TFM	Temps de fixation moyen.
ZI	Zone d'intérêt.
ZIP	Zone d'intérêt pertinente.

Glossaire

Acuité visuelle. Faculté qui permet de distinguer nettement, de voir séparément 2 points, 2 lignes, 2 objets. *page 55*

Analyse de la variance. Test d'hypothèse, aussi appelé Anova, permettant d'évaluer l'impact d'une ou plusieurs variables indépendantes sur la valeur d'une variable dépendante. Il peut également être étendu pour prendre en compte plusieurs variables dépendantes. *page 32*

Bloc-notes spatio-visuel. Dans la théorie «Vision-Compréhension», élément de la mémoire de travail qui stocke les informations visuelles et spatiales. Il est plus rapide que la mémoire à long terme. *page 7*

Blocage. Lors d'une étude empirique, principe selon lequel on bloque les effets d'un facteur indésiré. *page 21*

Boucle articulatoire. Dans la théorie «Vision-Compréhension», élément de la mémoire de travail qui stocke les informations verbales. Il est plus rapide que la mémoire à long terme. *page 7*

Cible. En oculométrie, élément du *stimulus* que le sujet doit trouver afin de remplir sa tâche. *page 58*

Compréhension logicielle. Activité d'un programmeur engagé dans un processus de compréhension du fonctionnement d'un logiciel. *page 1*

Couplage. En programmation orientée-objet, lors d'une modification d'un composant, quantité de changements relatifs aux composants autres que celui ayant reçu la modification initiale. *page 44*

Définition. Lors de la planification d'une expérience, activité consistant à définir clairement les objectifs de l'expérience. *page 18*

Densité spatiale. Mesure de la couverture d'un *stimulus* par un parcours oculaire. *page 64*

Direction centrale. Dans la théorie «Vision-Compréhension», processus faisant partie de la mémoire de travail qui exécute les tâches cognitives. *page 7*

Distance de Hausdorff. Mesure de la distance séparant 2 ensembles finis, de valeur définie par : $h(A,B) = \max\{\max_{a\in A}\min_{b\in B} d(a,b), \max_{b\in B}\min_{a\in A} d(b,a)\}$. *page 36*

Distribution normale ou loi normale. Distribution de probabilité théorique la plus utilisée. Elle suit la densité suivante : $f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$, avec σ l'écart-type et μ la moyenne. *page 28*

Distribution normale standard. Distribution de probabilité normale de moyenne 0 et d'écart-type 1. *page 28*

Distribution de probabilité théorique ou loi de probabilité. Définit la fréquence des valeurs possibles pour les points de la distribution. *page 27*

Échantillonnage. Activité de sélection d'un échantillon d'une population. *page 21*

Échelle d'intervalle. Famille d'échelles pour laquelle il existe une relation d'ordre et une notion de différence entre 2 mesures. *page 15*

Échelle de rapport. Famille d'échelles pour laquelle il existe une relation d'ordre, une notion de différence entre 2 mesures et pour laquelle le rapport entre 2 mesures a un sens. *page 15*

Échelle nominale. Famille d'échelles faisant simplement correspondre à l'attribut mesuré un nom ou un symbole. *page 15*

Échelle ordinale. Famille d'échelles pour laquelle il existe une relation d'ordre. *page 15*

En ligne. Une expérience en ligne est une expérience qui se déroule sur le terrain dans les conditions normales d'apparition des phénomènes. *page 13*

Équilibrage. Lors d'une étude empirique, principe selon lequel on donne à plusieurs groupes des tailles relativement équivalentes. *page 21*

Erreur de type I. Lors d'un test d'hypothèse, probabilité de rejeter H_0 par erreur (voir aussi Erreur de type II et Puissance d'un test statistique). *page 30*

Erreur de type II. Lors d'un test d'hypothèse, probabilité de ne pas rejeter H_0 par erreur (voir aussi Erreur de type I et Puissance d'un test statistique). *page 30*

Étude primaire. Étude récoltant des données brutes et fournissant une connaissance de base. *page 15*

Étude secondaire. Étude synthétisant les connaissances de base existantes. *page 15*

Facteur. Lors d'une expérience, variable indépendante dont l'influence est étudiée et dont la valeur n'est pas fixée mais contrôlée. *page 17*

Facteur confondant. Lors d'une étude empirique, facteur qui rend impossible la distinction des effets d'un facteur par rapport à un ou plusieurs autres. Lors d'une expérimentation, ces facteurs introduisent un biais dans l'analyse des liens entre les variables. *page 13*

Fixation. Stabilisation de la rétine de l'œil sur un objet. *page 56*

Forme canonique. Agencement des classes d'un patron de conception sur un diagramme de classes UML respectant celui utilisé par le GoF dans [GHJV95]. *page 53*

Fovea. Zone de l'œil avec la meilleure acuité visuelle. *page 55*

Gang of Four. Surnom donné aux 4 auteurs de l'ouvrage pionnier des patrons de conception : Erich Gamma, Richard Helm, Ralph Johnson et John Vlissides. *page 41*

Génie logiciel. (1) Application d'une approche systématique, disciplinée et quantifiable aux étapes de développement, d'opération et de maintenance de logiciel ; c'est l'application de l'ingénierie au logiciel. (2) Étude des approches telles que celles définies en (1). *page 10*

Génie logiciel empirique. Approche du génie logiciel utilisant des méthodes empiriques pour ses études. *page 10*

Graphique quantile-quantile. Graphique permettant de visualiser facilement la conformité de la distribution des valeurs d'un ensemble avec une distribution théorique. *page 31*

Homoscédasticité. Uniformité de la variance dans un ensemble de valeurs observées. *page 33*

Hors ligne. Une expérience hors ligne est une expérience qui se déroule dans un laboratoire où les conditions sont contrôlées et où les événements sont organisés afin de simuler les conditions d'apparition des phénomènes dans le monde réel. *page 13*

Hypothèse alternative. Lors d'un test d'hypothèse, il s'agit de l'hypothèse que l'on pense valide et que l'on veut tester. *page 29*

Hypothèse nulle. Lors d'un test d'hypothèse, il s'agit de l'hypothèse que l'on va tenter de réfuter au profit d'une ou de plusieurs hypothèses alternatives. *page 29*

Instrumentation. Lors de la planification d'une expérience, étape consistant à définir l'ensemble des besoins physiques de l'expérience. *page 22*

Interface. Dans le cadre des patrons de conception, toute classe ne pouvant pas être instanciée directement mais pouvant fournir un comportement par défaut. Correspond aux interfaces et classes abstraites de Java. *page 44*

Matrice de transition. Mesure de la recherche effectuée par un sujet sur un *stimulus*. *page 65*

Matrice de dissimilarité. Matrice $n \times n$ reprenant les mesures de dissimilarité par paires pour n objets. *page 37*

Matrice de proximité. Matrice de similarité ou de dissimilarité. *page 38*

Matrice de similarité. Matrice $n \times n$ reprenant les mesures de similarité par paires pour n objets. *page 38*

Mémoire à long terme. Dans la théorie «Vision-Compréhension», élément interagissant avec la mémoire de travail composé de : la mémoire épisodique, la mémoire procédurale et la mémoire sémantique. *page 7*

Mémoire de travail. Dans la théorie «Vision-Compréhension», mémoire similaire à la mémoire à court terme mais dont la structure, plus importante, est composée de : la direction centrale, le bloc-notes spatio-visuel, la boucle articulatoire et la mémoire à long-terme. *page 7*

Mémoire épisodique. Dans la théorie «Vision-Compréhension», élément de la mémoire à long terme qui stocke les informations sur les objets et une partie des événements de l'histoire de la vie du programmeur. Elle stocke les connaissances du domaine, les connaissances fonctionnelles acquises au cours de son apprentissage et les connaissances du logiciel acquises durant l'activité de compréhension. *page 7*

Mémoire procédurale. Dans la théorie «Vision-Compréhension», élément de la mémoire à long terme qui stocke les informations sur les compétences et les procédures pour exécuter les actions. Elle contient les connaissances de programmation et les stratégies pour la compréhension logicielle. *page 7*

Mémoire sémantique. Dans la théorie «Vision-Compréhension», élément de la mémoire à long terme qui stocke les informations relatives à la connaissance des concepts, comme dans une encyclopédie. Elle stocke les apparences visuelles des prototypes et catégories d'objets comme les architectures, les conceptions et les implémentations des prototypes. Elle stocke aussi des connaissances du domaine (acquises par la vue ou construites par la direction centrale). *page 7*

Mesure de dissimilarité. Mesure de l'éloignement entre 2 objets. *page 37*

Mesure de proximité. Mesure de la similarité ou de la dissimilarité entre 2 objets. *page 37*

Mesure de similarité. Mesure de la ressemblance entre 2 objets. *page 38*

Mesure valide. Une mesure est dite valide lorsqu'elle possède certaines caractéristiques par rapport à l'attribut qu'elle mesure :
- Elle doit être une caractérisation mathématique de l'attribut.
- Elle doit permettre la distinction entre 2 objets différents mais, dans la limite de l'erreur de mesure, des objets doivent pouvoir avoir la même valeur.
- Elle doit préserver notre notion intuitive de l'attribut et la manière avec laquelle nous différencions différents objets.

page 15

Méthode agglomérative. Méthode hiérarchique de classification procédant par assemblage de groupes existants. *page 38*

Méthode analytique. En recherche, méthode selon laquelle une théorie formelle est proposée et ensuite comparée expérimentalement. Cette méthode est traditionnellement utilisée dans des domaines plus formels tels que le génie électrique. *page 12*

Méthode divisive. Méthode hiérarchique de classification procédant par division de groupes existants. *page 38*

Méthode empirique. En recherche, méthode selon laquelle un modèle est proposé et évalué via des études empiriques. Cette méthode était traditionnellement utilisée en psychologie et dans les sciences sociales où il est impossible de se baser sur des lois de la nature comme en physique. C'est le comportement humain qui est étudié ici. *page 12*

Méthode hiérarchique. Méthode de classification ne nécessitant pas la définition préalable d'un nombre de groupes voulu. *page 38*

Méthode ingénieurale. En recherche, méthode selon laquelle les solutions du moment sont étudiées, des changements sont proposés et ensuite évalués. C'est probablement la méthode la plus utilisée par l'industrie. *page 11*

Méthode de partitionnement. Méthode de classification nécessitant la définition préalable d'un nombre de groupes voulu. *page 38*

Méthode scientifique. En recherche, méthode selon laquelle un modèle est construit sur base d'observations du monde. Cette méthode est traditionnellement utilisée dans les domaines appliqués. *page 11*

Mouvement de vergence. Mouvement de l'œil ajustant l'orientation des yeux en fonction de la distance de l'objet en profondeur. Ils permettent la perception de la perspective. *page 55*

Mouvement optocinétique. Mouvement de l'œil maintenant les images stables sur la rétine durant les rotations lentes et prolongées de la tête. *page 55*

Mouvement de poursuite lente. Mouvement de l'œil suivant un objet allant de $40^o/s$ à $100^o/s$ environ. *page 56*

Mouvement vestibulo-oculaire. Mouvement de l'œil maintenant les images stables sur la rétine durant les rotations brèves et rapides de la tête. *page 55*

Objet. Lors d'une expérience, support sur lequel le sujet exécute sa tâche. *page 17*

Oculomètre. Dispositif servant à enregistrer les mouvements oculaires d'une personne. *page 54*

Oculométrie. Technique d'enregistrement du parcours oculaire d'une personne. *page 53*

P-valeur. Une p-valeur exprime la probabilité qu'ont les résultats d'un test d'hypothèse d'être considérés comme significatifs alors qu'ils ne le sont pas. *page 30*

Parcours oculaire. En oculométrie, séquence «fixation – saccade – fixation – etc.» ordonnée chronologiquement. *page 57*

Patron de conception ou «design pattern». D'une manière générale, techniques de conception de logiciels orientés-objets éprouvées. *page 41*

Planification. Lors de la réalisation d'une expérience, activité consistant à définir «comment» l'expérience va être réalisée. *page 19*

Population. Ensemble des entités pour lesquelles le statisticien aimerait généraliser ses conclusions. *page 20*

Prototype. Dans la théorie «Vision-Compréhension», exemple d'objet représentatif d'une catégorie d'objets. *page 7*

Puissance d'un test statistique. Lors d'un test d'hypothèse, probabilité de rejeter H_0 correctement (voir aussi Erreur de type I et Erreur de type II). *page 30*

Quasi-expérience. Il s'agit d'une expérience pour laquelle il est impossible d'assigner aléatoirement les sujets aux traitements. *page 13*

Randomisation. Principe selon lequel une allocation se fait de manière aléatoire. *page 21*

Rapport de fixation (on target all). Mesure de la pertinence de l'effort d'un sujet concernant un *stimulus*. *page 65*

Réplication. En génie logiciel expérimental, une réplication consiste à reproduire une étude en utilisant la même conception. On parlera de «vraie» réplication dans le cas où les résultats sont également reproduits. *page 14*

Saccade. Mouvement rapide de l'œil utilisé pour positionner la fovea à un nouvel endroit de l'environnement visuel. *page 55*

Statistique inférentielle. Technique statistique permettant de prendre une décision concernant une population sur base de l'étude d'un échantillon de celle-ci. *page 27*

***Stimulus*.** En oculométrie, objet visualisé par un sujet. *page 54*

Sujet. Lors d'une expérience, personne qui participe à une expérience afin d'évaluer un objet. *page 17*

Temps de fixation moyen. Mesure de la charge cognitive nécessaire à l'interprétation d'un objet. *page 65*

Test bilatéral. Lors d'un test d'hypothèse, on considère les 2 hypothèses alternatives possibles : $<$ et $>$. Généralement résumé en $\neq$ (voir aussi Test unilatéral). *page 30*

Test d'hypothèse. Technique statistique permettant de se positionner sur un problème de la vie réelle concernant une population en se basant sur l'observation d'un échantillon de celle-ci. *page 29*

Test non-paramétrique. En statistique inférentielle, test d'hypothèse n'ayant pas de conditions sur l'ensemble des données. *page 31*

Test de normalité. Test d'hypothèse permettant d'évaluer si les données d'un échantillon respectent une distribution normale. *page 31*

Test paramétrique. En statistique inférentielle, test d'hypothèse ayant certaines conditions sur l'ensemble des données. *page 31*

Test unilatéral. Lors d'un test d'hypothèse, l'on ne considère qu'une seule des 2 hypothèses alternatives possibles : soit $<$ soit $>$ (voir aussi Test bilatéral.). *page 29*

Théorème central limite. Théorème établissant que la distribution des valeurs d'un échantillon d'une population est une approximation de la distribution normale, quelle que soit la population à l'origine de l'échantillon, pour peu que l'échantillon soit suffisamment grand. *page 29*

Théorie «Vision-Compréhension». Théorie combinant les théories de la compréhension logicielle et les théories de la science de la vision pour expliquer l'acquisition des données lors de l'activité de compréhension logicielle. *page 5*

Traitement. Lors d'une expérience, valeur particulière donnée à un facteur. *page 17*

Transformation d'échelle admissible. On parle de transformation admissible lorsque l'on transforme une mesure définie sur un type d'échelle en une autre mesure et que les relations entre objets sont préservées. *page 15*

Transformation de données. En statistique, activité consistant à effectuer une transformation valide des données afin de leur faire respecter certaines propriétés. *page 31*

Valeur significative. Lors d'un test d'hypothèse, se dit lorsque la p-valeur se situe sous un seuil préalablement déterminé. *page 30*

Valeur solitaire ou extrême. En statistique descriptive, dans un ensemble de valeurs, valeur largement à l'écart des autres ; elle est beaucoup plus grande (resp. petite) que l'ensemble des autres. *page 25*

Validation des données. Lors d'une expérimentation, activité prenant place après la récolte des données. Elle sert à vérifier que les données sont cohérentes. *page 24*

Validité de conclusion. Lors d'une expérience, validité statistique de la relation entre le traitement et le résultat. *page 22*

Validité de construction. Lors d'une expérience, validité de la correspondance entre la théorie et les observations. *page 22*

Validité externe. Lors d'une expérience, validité de la généralisation des résultats obtenus sur l'échantillon à la population étudiée. *page 22*

Validité interne. Lors d'une expérience, validité de la relation réelle entre le traitement et le résultat. *page 22*

Variable dépendante. Lors d'une expérience, variable pour laquelle l'expérimentateur désire étudier le changement provoqué par la manipulation des variables indépendantes. Elle peut être de 3 types : de référence, intermédiaire ou de réponse. *page 17*

Variable indépendante. Lors d'une expérience, variable soit manipulée, soit contrôlée par l'expérimentateur. *page 17*

Variable intermédiaire. Lors d'une expérience, variable permettant d'expliquer la relation de cause à effet observée dans le cas où la variable de réponse ne peut pas le faire directement. *page 20*

Variable de référence. Lors d'une expérience, variable qui permet de mesurer les différences induites par un traitement. *page 20*

Variable de réponse. Lors d'une expérience, variable permettant de vérifier l'existence d'une relation de cause à effet. *page 20*

Zone d'intérêt. En oculométrie, zone d'un environnement visuel jugée intéressante par le(s) chercheur(s) (*e.g.* les classes et relations dans un diagramme de classe UML). *page 58*

Zone d'intérêt pertinente. En oculométrie, zone d'un environnement visuel jugée particulièrement intéressante par le(s) chercheur(s) (*e.g.* les classes et relations dans un diagramme de classe UML contenant une information utile à la réalisation de la tâche demandée). *page 58*

Chapitre 1

Introduction

Le processus de création de logiciels inclut un grand nombre d'activités. Ce processus comprend l'analyse des exigences, la conception de l'architecture, l'implémentation, le test, la maintenance, etc. Différentes approches ont été proposées dans la littérature pour organiser ces activités : modèle en cascade, modèle en spirale, modèle agile, etc. Si les manières d'organiser la conception d'un logiciel sont différentes, elles passent pratiquement toutes par les mêmes activités. L'activité la plus coûteuse de ce processus est généralement l'activité de maintenance. Cette activité de maintenance est elle-même composée de plusieurs activités comme la détection et la correction de défauts, l'ajout de nouvelles fonctionnalités, etc. La pratique a montré que les programmeurs travaillant à la maintenance d'un logiciel ne sont généralement pas ceux qui en sont à l'origine. Ainsi, le « nouveau » programmeur doit, avant de commencer son travail de maintenance, passer par une période de compréhension du logiciel qu'il doit maintenir. Pour ce faire, il va consulter les éléments utiles à la compréhension (*e.g.*, la documentation, le code source, etc.) qu'il a à sa disposition. Cette activité que nous nommons *compréhension logicielle* est une activité de base pour tout programmeur.

Plusieurs théories [Bro78, vMV95] tentent d'expliquer ce processus de compréhension logicielle mais aucune ne reconnaît explicitement l'importance de la vue dans cette activité. La théorie « Vision-Compréhension » [Gué05] tente de combler ce manque et propose une jonction des théories précédemment citées avec les théories de la science de la vision pour expliquer le processus d'acquisition des informations nécessaires au programmeur lors de son activité de compréhension logicielle.

Guéhéneuc, l'auteur de la théorie « Vision-Compréhension », propose plusieurs exemples d'applications de sa théorie pour expliquer des faits bien connus en génie logiciel. Il utilise sa théorie pour expliquer le fulgurant intérêt apporté aux patrons de conception [GHJV95]. Ces patrons de conception sont réputés pour leurs qualités (*e.g.* flexibilité, ré-utilisabilité, etc.) et reconnus pour faciliter la communication. Ainsi, deux personnes connaissant un même patron de conception peuvent en parler comme d'un élément de base, plutôt que d'en parler à partir de chacune des classes qui le composent. De la même manière, un programmeur impliqué dans une activité de compréhension logicielle détecterait plus facilement les patrons de conception qu'il connaît par rapport à un programmeur ne les connaissant pas. Une fois le patron de conception détecté, le programmeur identifierait également ses fonctionnalités plus facilement. Cette qualité des patrons de conception n'était explicable par aucune théorie mais, surtout, n'a que rarement été étudiée de manière empirique.

L'oculométrie est une technique qui enregistre les coordonnées du regard d'une personne pendant une activité mentale. Cette discipline permet d'analyser le parcours oculaire d'une personne et d'identifier les zones sur lesquelles elle porte son intérêt. Cette technique est utilisée depuis près de 30 ans par les sciences cognitives [Ray98]. Elle est également utilisée pour l'étude des interfaces

Homme–Machine, pour le marketing, pour la recherche médicale, pour évaluer l'attention d'un pilote de ligne, etc. Il existe néanmoins un frein majeur à l'adoption à grande échelle de cette technique : la définition des mesures et leur interprétation [JKD02].

Ce mémoire a pour objet, premièrement, l'étude des éléments et disciplines qui touchent à la réalisation d'une expérience en génie logiciel utilisant des représentations sous forme de diagrammes de classes UML ainsi que la technique de l'oculométrie et, deuxièmement, le compte-rendu d'une expérience que nous avons réalisée pour tenter d'apporter un élément de réponse concernant la validité de l'application de la théorie « Vision-Compréhension » aux patrons de conception.

En analysant le regard d'un programmeur expert par rapport à celui d'un programmeur novice sur un diagramme de classes UML, nous devrions être capables de déterminer l'objet précis de leur attention sur le diagramme. Nous pourrions dès lors vérifier si le programmeur expert détecte plus facilement les patrons de conception et s'il les utilise plus efficacement que le programmeur novice. Si tel est le cas, l'expérience vérifierait le fait généralement accepté que les patrons de conception aident à la compréhension logicielle et apporterait un élément en faveur de l'application de la théorie « Vision-Compréhension » aux patrons de conception. Dans la négative, cette expérience apporterait un élément à l'encontre de cette qualité supposée des patrons de conception et aussi à l'encontre de l'application de la théorie « Vision-Compréhension », ce qui permettrait de réviser et de proposer de nouvelles hypothèses à cette théorie.

Ce mémoire s'inscrit dans un contexte de génie logiciel empirique. Nous commençons par exposer la théorie « Vision-Compréhension » avant de proposer une étude de chacune des disciplines touchées par l'expérience que nous avons réalisée. Ces éléments permettent au lecteur de comprendre concrètement les difficultés de l'élaboration d'une expérience en génie logiciel, tant il y a de paramètres à prendre en compte. Suivent la description de la réalisation de l'expérience, son exécution pratique, l'analyse des données récoltées et enfin, l'analyse des menaces qui pèsent sur la validité de notre expérience.

Voici un bref résumé de chacun des chapitres de ce mémoire :

- Le **chapitre 1** introduit le mémoire et présente le contexte dans lequel il se situe.
- Le **chapitre 2** expose la problématique en détail. La théorie « Vision-Compréhension » est présentée ainsi que son application aux patrons de conception.
- Le **chapitre 3** explique ce que l'on entend par génie logiciel empirique et présente en détail le processus lié à la méthode expérimentale utilisée pour notre expérience.
- Le **chapitre 4** présente la statistique inférentielle utilisée pour l'analyse des données récoltées. Ce chapitre présente ensuite plus particulièrement l'interprétation des résultats au test d'hypothèse Anova ainsi que 2 concepts utilisés pour la classification des participants.
- Le **chapitre 5** revient sur la notion de patron de conception et introduit les 2 patrons étudiés dans notre expérience.
- Le **chapitre 6** traite des représentations graphiques mais ne revient pas sur la notation UML supposée connue et propose un tour d'horizon de la technique de l'oculométrie.
- Le **chapitre 7** définit formellement l'expérience, explique chacun des choix effectués lors de la conception de l'expérience et en décrit l'exécution pratique.
- Le **chapitre 8** présente l'analyse des données : la réduction de l'ensemble des données, la classification des participants ainsi que l'étude des hypothèses principales et auxiliaires.
- Le **chapitre 9** propose une analyse détaillée des menaces à la validité de l'expérience.
- Le **chapitre 10** conclut le mémoire et propose quelques pistes pour les travaux futurs.

D'une manière générale, le lecteur pressé est invité à consulter les parties suivantes : le chapitre 2 exposant la problématique ; la section 4.2, si nécessaire, expliquant la manière d'interpréter les résultats du test statistique utilisé (2-way Anova) ; si le lecteur n'est pas familier avec les patrons de conception Observateur ou Objet composite, la section 5.4 ; la section 6.2 concernant l'oculométrie ; enfin, concernant l'expérience, la section 7.2 présentant les choix effectués et plus particulièrement les hypothèses et variables utilisées et le chapitre 8 présentant l'analyse des données et plus particulièrement la section 8.6 résumant les résultats obtenus. Un glossaire et la liste des acronymes sont proposés en début de mémoire.

Chapitre 2

Problématique

Il existe plusieurs théories cognitives qui expliquent le processus de traitement des informations par le cerveau lors de la compréhension logicielle par un programmeur. Mais ces théories n'expliquent pas le processus d'acquisition de l'information par le programmeur. Guéhéneuc, dans un article intitulé « A Theory of Program Comprehension – Joining Vision Science and Program comprehension » [Gué05], propose la théorie « Vision-Compréhension » capable d'expliquer cette acquisition. Cette théorie, qui se base sur la science de la vision et sur la compréhension logicielle, se trouve à la base de notre travail.

La section 2.1 explique l'importance des théories pour la recherche. La section 2.2 motive la théorie « Vision-Compréhension » par rapport aux théories antérieures. La section 2.3 explique cette théorie. La section 2.4 en donne un exemple d'application aux patrons de conception. Enfin, la section 2.5 explique l'importance de la réalisation d'expériences afin de tester la validité de la théorie et présente le but de notre expérience. Ce chapitre est tiré de [Gué05], le lecteur désirant un complément d'information peut s'y référer.

2.1 Les théories et leur importance

> *A theory is an integrated set of statements – hypotheses – about the principles that organises and explains known facts and that makes predictions about new facts possible. [...] A theory helps in understanding a domain [...] [Gué05].*
>
> *[Une théorie est un ensemble intégré d'affirmations – d'hypothèses – sur des principes, qui organise et explique des faits connus et qui permet de faire des prédictions pour de nouveaux faits. [...] Une théorie aide à comprendre un domaine[...]]*

Théories en génie logiciel

Le génie logiciel propose des théories mais la jeunesse du domaine, la complexité des phénomènes et le manque de consensus concernant les formalismes et les notations font qu'elles n'expliquent pas des faits bien connus et qu'elles manquent de généralité car elles n'expliquent que quelques faits à la fois. Elles ressemblent plus à des lois qu'à des théories. De plus, elles ne sont que rarement validées.

Les études empiriques sont essentielles à la compréhension des phénomènes survenant en génie logiciel. Ces études sont basées sur un cycle classique : observation des faits, lois, théories. Actuellement, ces études ont permis d'enregistrer beaucoup de faits et quelques lois en ont découlé

mais il n'existe pas beaucoup de théories dérivées les expliquant et permettant la prédiction de nouveaux faits.

Il y a aussi un manque de cadres de référence facilitant la mise en œuvre et l'interprétation des résultats d'expériences. Les théories sont aussi une aide précieuse dans l'élaboration d'expériences observant les faits et permettant d'appuyer ou réfuter des lois ou théories en génie logiciel.

Théories en compréhension logicielle

Le domaine de la compréhension logicielle se compose de 3 sous-domaines. Le premier est l'acquisition des données à propos des logiciels. Cette acquisition peut se faire à partir de plusieurs sources : données statiques ou dynamiques, fonctionnalités, documentation et dépôts. Ce domaine étudie la représentation des données, leur communication, etc. Il est important pour la compréhension des données dont les programmeurs disposent.

Le second sous-domaine étudie les contextes dans lesquels l'activité de compréhension logicielle se déroule. Bien que cela ne soit pas directement lié à la compréhension logicielle, cela donne un aperçu de l'activité de compréhension logicielle qui est une activité de base de tout programmeur. Cette étude est importante pour la généralisation des résultats car cette généralisation n'est pas évidente.

Le dernier sous-domaine est le développement de théories basées sur la compréhension de l'utilisation des données extraites par le programmeur. Peu de théories ont été proposées mais Guéhéneuc se base néanmoins sur 2 d'entre elles : Brooks [Bro78] décrit le processus de la compréhension logicielle en tant que tel tandis que von Mayrhauser [vMV95] décrit le processus prenant place dans l'esprit du programmeur. Cependant, aucune de ces théories n'explique le processus d'acquisition des données.

Théorie en science de la vision

La science de la vision est le domaine qui s'intéresse au système de vision. Ses théories ont permis aux scientifiques de prédire avec succès de nouveaux faits et d'affiner leurs théories. Il en existe un grand nombre [Pal99] expliquant par exemple la vision en couleur, etc.

2.2 Motivation de la théorie

La théorie « Vision-Compréhension » se base sur le fait que les programmeurs sont des êtres humains. Leurs caractéristiques physiques et cognitives sont importantes car ils utilisent tous leurs sens et capacités cognitives, en particulier la vue, lorsqu'ils tentent de comprendre un logiciel. L'activité de compréhension nécessite la lecture de documentation, de code source et la visualisation de toutes sortes de modélisations. L'omniprésence de la vue est corroborée par la vaste littérature sur la visualisation pour la compréhension logicielle. Malgré cela, aucune théorie sur la compréhension logicielle ne reconnaît explicitement l'importance de la vue.

La théorie « Vision-Compréhension » proposée étend celles proposées par Brooks [Bro78] et von Mayrhauser [vMV95], en joignant à ces théories de la compréhension logicielle la théorie de la science de la vision, afin de prendre en compte l'acquisition et l'utilisation de l'information visuelle dans la compréhension logicielle par le programmeur.

En plus d'expliquer des faits connus et d'en permettre la prédiction de nouveaux, cette théorie permettrait de diriger les efforts visant à automatiser (partiellement) l'activité de compréhension logicielle. De plus, elle permettrait d'aider à l'élaboration d'expériences visant à répondre à des questions relatives à l'activité de compréhension logicielle par la vision.

2.3 La théorie « Vision-Compréhension »

2.3.1 Introduction

L'activité de compréhension logicielle requiert plus que l'information contenue dans le code source et la documentation. La tâche à accomplir (pourquoi), le contexte de l'activité (où) et l'expérience du programmeur (avec le génie logiciel, la compréhension et le logiciel étudié) sont des sources d'information additionnelles. Le programmeur lui-même amène de l'information durant l'activité de compréhension logicielle.

La théorie « Vison-Compréhension » se positionne dans le paradigme du traitement de l'information pour lequel le cerveau humain est vu comme une machine à calcul. Ce paradigme est fondé sur la similitude entre la psychologie cognitive et l'informatique. La théorie fait l'hypothèse que les processus cognitifs tels que la perception visuelle et la compréhension logicielle sont des processus de transformation d'information, prise en entrée, en une autre information, fournie en sortie. Ils peuvent être décomposés en plus petits processus cognitifs liés par un flux et sont incarnés par le comportement physique du cerveau humain où ils prennent place. Supposant que le programmeur focalise son attention sur la modélisation du logiciel, la théorie étudie et décrit la perception visuelle ainsi que l'activité de compréhension par des processus agissant sur différentes représentations de l'information tirée de la modélisation du logiciel. Une théorie de l'apprentissage par renforcement est également considérée pour expliquer le comportement de certains processus par rapport à la quantité de traitement nécessaire à l'exécution de l'activité de compréhension logicielle.

Le champ d'application de la théorie est limité au programmeur engagé dans une activité de compréhension logicielle. Le programmeur doit utiliser sa vision dans des conditions normales et les modélisations peuvent être sous différentes formes : textuelles, graphiques, statiques, dynamiques, etc.

2.3.2 Définition

Pour la définition de la théorie, nous appelons « représentation » une représentation cognitive interne d'un logiciel et « modèle » une modélisation externe d'un logiciel comme le code source ou un diagramme UML.

La reconnaissance d'objets au cours de la compréhension d'un logiciel est décomposée en plusieurs processus agissant sur les différentes représentations d'un modèle du logiciel. La figure 2.1 montre la séquence des processus et du flux de données.

Image rétinienne : l'activité de compréhension logicielle commence avec une image rétinienne d'une modélisation d'un logiciel observée par le programmeur.

Étape basée sur l'image : l'image rétinienne est analysée pour en extraire les traits caractéristiques tels que les arêtes, lignes et textures. Ce processus produit un ensemble d'éléments visuels : arêtes, barres et taches.

Étape basée sur les surfaces : les éléments visuels du processus précédent sont interprétés en terme de surfaces visibles dans un espace à 3 dimensions. Le processus produit des bouts de surface orientés.

Étape basée sur les objets : ce processus concerne la perception de l'organisation des éléments visuels (bouts locaux de surface orientés inclus) en objets. Ce processus est décomposé en 4 sous-processus :

Analyse des régions : ce sous-processus analyse les éléments visuels pour identifier les régions dans la structure de l'image. Une région est une zone délimitée à 2 dimensions,

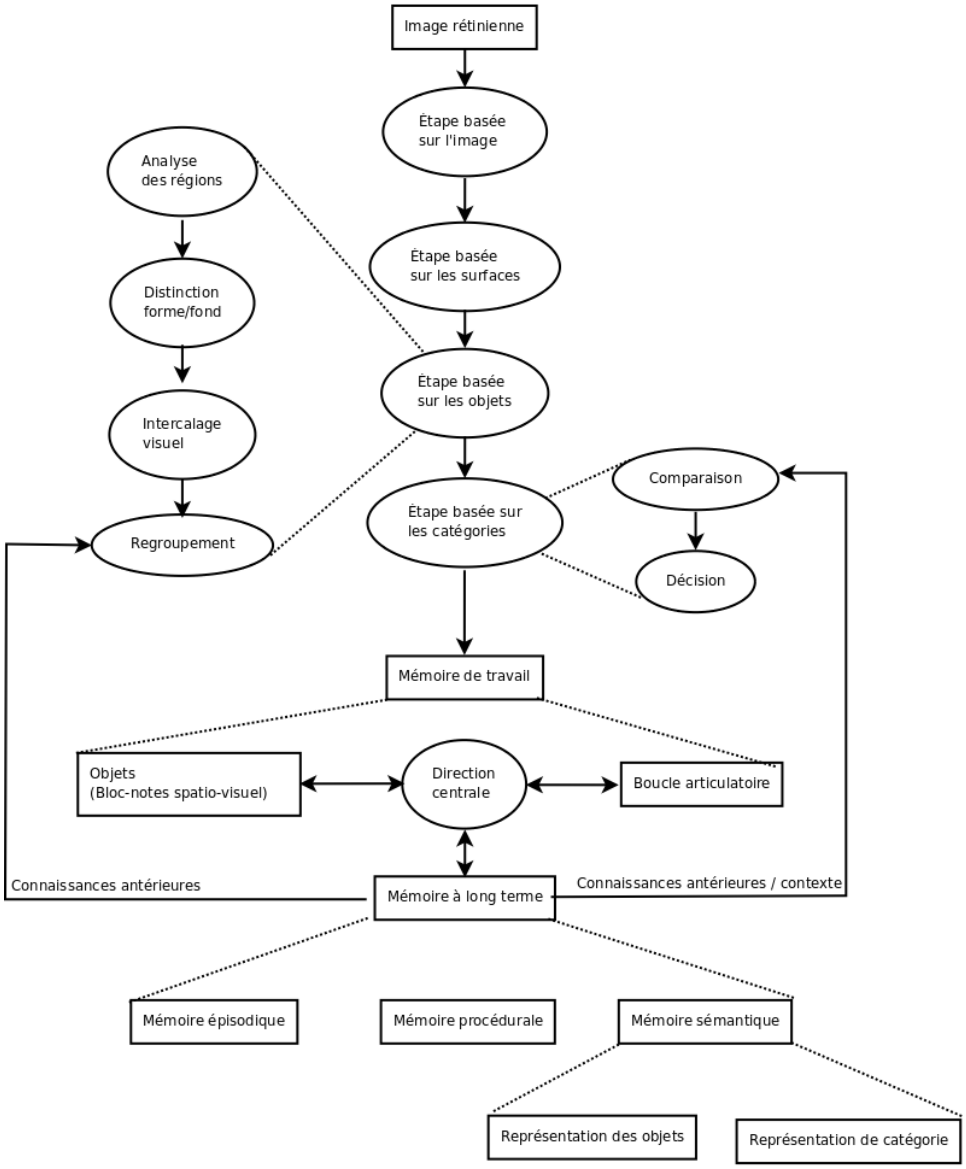


FIG. 2.1: Théorie de la compréhension logicielle par la vision (adaptée de [Gué05]). Les boîtes représentent les représentations d'une image, les cercles sont les processus, les flèches décrivent le flux d'information, les lignes pointillées montrent les décompositions. L'absence de représentation entre les processus n'en exclut pas l'existence mais simplifie le flux.

qui constitue un sous-ensemble de l'image. Différents principes dirigent l'analyse des régions, tels que la connexité, la segmentation ou l'isolement des textures.

Distinction figure/fond : une fois les régions identifiées, ce sous-processus décompose les régions en 2 catégories : les figures et le fond. Les figures sont des parties de l'image proches de l'observateur, elles sont entourées par un contour définissant leur forme. Le fond est la partie de l'image plus éloignée de l'observateur, s'étendant à l'arrière et non délimitée par un contour.

Intercalage visuel : ce sous-processus intercale les parties cachées des régions identifiées

à partir de leurs parties visibles afin de permettre au système visuel de percevoir les éléments visuels partiellement représentés.

Regroupement : ce sous-processus groupe les éléments visuels précédemment identifiés en objets cohérents. Il utilise les lois de regroupement, telles que la proximité, les similitudes de couleur, la taille ou l'orientation. Il utilise également les connaissances antérieures, stockées dans la mémoire à long terme, pour effectuer les regroupements.

Étape basée sur la catégorie : grouper les éléments visuels en objets n'est pas suffisant pour permettre la compréhension. Les objets doivent être classés pour distinguer leurs fonctions. La théorie suit l'approche théorique de la perception indirecte de la fonction, qui se décompose en 2 sous-processus : comparaison et décision. La théorie suppose que la mémoire stocke des exemples d'objets, représentés par des *prototypes* à partir desquels des catégories sont attribuées aux objets identifiés.

Comparaison : ce sous-processus utilise les connaissances antérieures et le contexte pour comparer les objets identifiés à des objets connus. Les objets peuvent appartenir à plusieurs catégories (ou à aucune).

Décision : ce sous-processus décide des catégories auxquelles appartiennent les objets.

Mémoire de travail : les processus précédents ont produit une représentation dans laquelle les différents éléments qui composent la modélisation du programme ont été identifiés et catégorisés. Cette représentation est prise en main par la mémoire de travail, qui est similaire à la mémoire à court terme, mais qui possède une structure interne plus importante.

Direction centrale : le processus de direction centrale exécute les tâches cognitives, telles que la compréhension, la résolution de problèmes ou la mémorisation des tâches. Elle travaille en étroite collaboration avec les 2 mémoires suivantes.

Bloc-notes spatio-visuel et boucle articulatoire : ce bloc-notes spatio-visuel stocke les informations visuelles/spatiales tandis que la boucle articulatoire stocke les informations verbales. D'autres types de mémoires peuvent exister pour d'autres modalités. Ces mémoires jouent le rôle de mémoire vive par rapport à la mémoire à long terme, étant plus rapides et plus faciles d'accès pour la direction centrale.

Mémoire à long terme : le processus de direction centrale interagit aussi avec la mémoire à long terme. La mémoire à long terme est composée de 3 mémoires : épisodique, procédurale et sémantique.

Mémoire épisodique : la mémoire épisodique stocke les informations sur les objets et une partie des événements de l'histoire de la vie du programmeur. Elle stocke les connaissances du domaine, les connaissances fonctionnelles acquises au cours de son apprentissage et les connaissances du logiciel acquises durant l'activité de compréhension.

Mémoire procédurale : la mémoire procédurale stocke les informations sur les compétences et les procédures pour exécuter les actions. Elle contient les connaissances de programmation et les stratégies pour la compréhension logicielle.

Mémoire sémantique : enfin, la mémoire sémantique stocke les informations relatives à la connaissance des concepts, comme dans une encyclopédie. Elle stocke les apparences visuelles des prototypes et catégories d'objets comme les architectures, les conceptions et les implémentations des prototypes. Elle stocke aussi des connaissances du domaine (acquises par la vue ou construites par la direction centrale).

La théorie présentée décrit l'activité de compréhension logicielle depuis une image rétinienne, générée à partir d'une représentation visuelle d'une modélisation d'un logiciel, jusqu'à la direction centrale qui exécute l'activité de compréhension.

2.4 Application aux patrons de conception

Les patrons de conception ont été rapidement et largement adoptés par la communauté du génie logiciel. En particulier, ceux décrits par Gamma *et al.* [GHJV95] ont suscité beaucoup d'intérêt dans la communauté, qui a vu en eux un moyen de documenter et de réutiliser les bonnes pratiques en matière de conception. L'abondante littérature prétend qu'ils facilitent la compréhension logicielle mais peu d'études empiriques ont tenté d'appuyer ou réfuter ces affirmations.

Les motifs de conception (solutions des patrons de conception) peuvent être vus comme des prototypes pouvant directement être utilisés par l'activité de compréhension de micro-architectures (sous-ensemble de la conception d'un logiciel). Les motifs de conception seraient proches voire identiques aux prototypes présents dans la mémoire du programmeur. Un programmeur connaissant ces motifs reconnaîtrait directement leurs structure et fonctionnalités. Si on prend la figure 5.5 (page 47), le programmeur expérimenté devrait reconnaître immédiatement la structure de l'objet composite (illustré par la figure 5.4, page 47) tandis qu'un programmeur novice ne verrait qu'une hiérarchie de 5 classes, quelques opérations et une relation d'agrégation. Trois cas sont à considérer :

Le programmeur ne connaît pas les patrons de conception : un programmeur sans connaissance des patrons de conception extrairait les composants de la micro-architecture via les étapes basées sur l'image et les surfaces. La mémoire à long terme ne pourrait pas aider les étapes basées sur les objets et les catégories avec ses connaissances antérieures. La mémoire de travail devrait dès lors faire une analyse des éléments, coûteuse en temps et en ressources, pour identifier les fonctions.

Le programmeur connaît un patron de conception autre que celui utilisé : ce cas est le même que celui présenté ci-dessus. L'information fournie par la mémoire à long terme n'aide pas.

Le programmeur connaît le patron de conception utilisé : la connaissance du patron de conception utilisé pour concevoir la micro-architecture diminue la charge de la mémoire de travail. Le programmeur extrairait les composants de la micro-architecture via les étapes basées sur l'image et la surface. Le sous-processus de regroupement grouperait ces éléments en utilisant la connaissance du prototype du motif de conception similaire. Le sous-processus de comparaison utiliserait ensuite cette information et la connaissance du prototype pour donner directement le fonctionnement de la micro-architecture à la mémoire de travail, facilitant la compréhension.

Ceci propose une explication à la rapide adoption des patrons de conception et l'attention donnée à leurs formes littéraires : la description synthétique des patrons de conception basée sur les prototypes est utilisable directement dans les étapes basées sur les objets et les catégories, lors de la compréhension logicielle par la vision, diminuant la charge de la mémoire de travail et facilitant l'activité de compréhension.

La théorie « Vision-Compréhension » suggère donc une explication à l'importance des patrons de conception : ils réduisent la quantité de traitement nécessaire à la direction centrale pour comprendre les fonctions d'un sous-ensemble d'une modélisation d'un logiciel. Elle fournit également une explication de l'avantage qu'a un expert en génie logiciel par rapport à un novice : l'expert a lu et compris une plus grande quantité d'information concernant l'implémentation de logiciels (code source) et de conception (modélisation telle qu'UML). Cela accroît le nombre de prototypes présents dans sa mémoire sémantique. Enfin, cette théorie souligne l'importance d'une bonne modélisation de logiciel graphique. Elles sont importantes car elles facilitent les processus de regroupement et de comparaison.

2.5 Falsification expérimentale

Afin d'affiner cette théorie, il est important de tenter de la falsifier (montrer ses défauts). Des études pourraient montrer qu'une partie (ou l'entièreté) de cette théorie est fausse, faisant ainsi également avancer la compréhension de l'activité de compréhension logicielle. C'est dans ce cadre que notre travail se situe.

La falsification expérimentale de cette théorie comprend l'élaboration d'expériences tentant de réfuter le flux de données entre les processus, l'utilisation des connaissances antérieures durant les sous-processus de regroupement et de comparaison, ou l'utilité de ces sous-processus pour la compréhension logicielle. Une possibilité pour mener ce type d'expérience est d'observer le programmeur engagé dans une activité de compréhension logicielle.

Les progrès concernant l'enregistrement du comportement humain de manière non intrusive permettent de suivre l'activité extérieure du programmeur. Les oculomètres permettent de suivre les mouvements oculaires. En enregistrant les mouvements oculaires d'un programmeur, nous devrions être en mesure d'étudier l'identification de patrons de conception et leur utilisation durant la compréhension logicielle.

Lors de notre expérience, nous allons utiliser un oculomètre pour vérifier que des novices n'identifient effectivement pas les patrons de conception dans une modélisation de micro-architecture. Les modélisations utilisées seront des diagrammes de classes UML. Le résultat attendu est que, contrairement aux novices, les experts identifient correctement les patrons de conception ; lors de la réalisation d'une tâche relative aux patrons, ils porteraient davantage leur attention sur les éléments utiles de la modélisation, et cette même tâche réalisée par les novices devrait requérir plus de temps/traitements, ceci en raison de la non pertinence de leurs efforts.

Chapitre 3

Génie logiciel expérimental

La réalisation d'une expérience suit une stratégie de recherche appelée expérimentale. Cette technique n'est qu'une stratégie parmi d'autres pour les études empiriques. La section 3.1 présente ce que l'on entend par génie logiciel empirique et présente les différentes stratégies. La section 3.2 présente en détail le processus de la stratégie expérimentale que nous avons suivie.

3.1 Génie logiciel empirique

Il est impossible aujourd'hui de se passer des logiciels. Ils sont omniprésents dans notre société et sont développés en masse. Ce développement est un processus créatif, impossible à automatiser complètement et à la source d'un grand nombre de problèmes, identifiés dès les années 60. Le génie logiciel est la discipline qui étudie le processus de développement des logiciels. L'IEEE [1] a défini une terminologie propre au génie logiciel dans [IEE90]. Le génie logiciel y est défini comme :

> *« (1) The application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software ; that is, the application of engineering to software. (2) The study of approaches as in (1). »*
>
> *« [(1) L'application d'une approche systématique, disciplinée et quantifiable aux étapes de développement, d'opération et de maintenance de logiciel ; c'est l'application de l'ingénierie au logiciel. (2) L'étude des approches telles que celles définies en (1).] »*

Trois aspects de cette définition sont importants dans le cadre du génie logiciel empirique :

1. La notion d'étape (cycle de vie) implique la présence d'un processus.
2. La définition insiste sur la nécessité d'une approche systématique et disciplinée.
3. L'importance de la quantification est soulignée.

Cette section a pour but de donner une présentation sommaire de ce qu'est le génie logiciel empirique. Nous commençons par présenter le contexte dans lequel il s'inscrit dans la section 3.1.1 pour ensuite, dans la section 3.1.2, expliquer pourquoi il faut considérer cette discipline comme une science. Nous donnons, dans la section 3.1.3, les principales stratégies empiriques avant d'aborder une première fois, avec la section 3.1.4, le domaine problématique des mesures. D'une manière générale, cette section est inspirée par l'ouvrage de Wohlin *et al.* « Experimentation in software engineering : An Introduction » [WRH+00].

[1] `http://www.ieee.org/portal/site`

3.1.1 Le contexte

Le processus de développement de logiciel décrit les étapes et activités nécessaires au développement d'un logiciel. La manière la plus simple de modéliser ce processus de développement est présentée par la figure 3.1. Les processus utilisés en réalité sont plus compliqués et bien connus des informaticiens. Il s'agit des modèles tels que le modèle en cascade, le modèle en spirale, le modèle agile, etc.

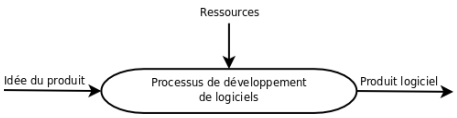


FIG. 3.1: Schématisation du processus de développement de logiciels.

Bien souvent, la complexité d'un logiciel fait que son développement est une tâche complexe, menée sur une longue période de temps et impliquant beaucoup de monde. De ce fait, le processus de développement, qui inclut le développement et la maintenance du logiciel, est lui aussi complexe. Or, un processus complexe n'est pas facile à mettre en place de façon efficace, c'est pourquoi l'industrie tente toujours de l'améliorer afin d'augmenter la qualité de ses produits et diminuer leur coût. Le génie logiciel met en évidence l'importance d'une approche systématique et disciplinée du travail. Cette approche est aussi nécessaire lors de l'amélioration du processus en lui-même. C'est pourquoi, bien souvent, un processus d'amélioration du processus est aussi mis en place. Il inclut principalement 2 activités :

L'évaluation du processus : elle est utile afin de déterminer où des améliorations sont nécessaires.

L'évaluation d'une proposition d'amélioration du processus : lorsqu'une possibilité d'amélioration est détectée et qu'une proposition d'amélioration est formulée, il est préférable d'évaluer la proposition avant de la mettre en pratique. C'est lors de cette étape que l'expérimentation prend son importance car elle permet d'évaluer des activités humaines.

L'expérimentation est également utile pour la recherche en génie logiciel en tant que discipline scientifique. Il ne suffit pas de proposer de nouvelles techniques ou de nouveaux langages, il faut également les comparer avec les anciennes techniques ou anciens langages.

3.1.2 Génie logiciel et science

Le génie logiciel est pluridisciplinaire. Il va des problèmes techniques, tels que les bases de données et systèmes d'exploitation, aux problèmes psychologiques en passant par les problèmes de langages tels que la syntaxe et la sémantique. Bien que le développement de logiciel soit une discipline basée sur la créativité humaine, il faut la traiter comme une science. Comme HABRA *et al.* [HALS08] le précise bien, chaque science moderne tente d'expliquer des phénomènes se produisant dans le monde physique sur base d'une modélisation de ceux-ci. Bien qu'un logiciel soit souvent vu comme un produit intellectuel, il s'agit également d'une modélisation particulière de phénomènes physiques se produisant dans un ordinateur et le génie logiciel doit à ce titre être considéré comme une discipline scientifique, ce qui implique l'utilisation de méthodes scientifiques.

Il est donc important de connaître et comprendre les méthodes de recherche disponibles, leurs limitations et leurs applicabilités. WOHLIN *et al.* [WRH+00] ressortent 4 méthodes de la littérature :

La méthode scientifique : un modèle est construit sur base d'observations du monde. Cette méthode est traditionnellement utilisée dans les domaines appliqués (*e.g.*, pour évaluer les performances d'un réseau de télécommunication).

La méthode ingénieurale : les solutions du moment sont étudiées, des changements sont proposés et ensuite évalués. C'est probablement la méthode la plus utilisée par l'industrie.

La méthode empirique : un modèle est proposé et évalué via des études empiriques (voir la section 3.1.3). Cette méthode était traditionnellement utilisée en psychologie et dans les sciences sociales où il est impossible de se baser sur des lois de la nature comme en physique. C'est le comportement humain qui est étudié ici. Et le génie logiciel est largement dominé par sa composante humaine : nous ne devons pas nous attendre à trouver des lois formelles générales.

La méthode analytique : une théorie formelle est proposée et ensuite comparée expérimentalement. Cette méthode est traditionnellement utilisée dans des domaines plus formels tels que le génie électrique.

Les 3 méthodes autres que la méthode empirique ne doivent pas être rejetées. Elles peuvent très bien convenir pour l'étude de certains aspects du génie logiciel, mais la forte composante humaine fait que c'est généralement la méthode empirique qui est utilisée. Le principal but de la méthode empirique est d'identifier et comprendre les relations entre différentes variables.

3.1.3 Stratégies empiriques

Commençons par préciser qu'une personne participant à une expérience afin d'évaluer un objet est appelée « sujet » et qu'un « objet » correspond par exemple à un document devant être évalué avec différentes techniques.

Wohlin *et al* [WRH+00] identifient dans la littérature 2 approches différentes correspondant à 2 paradigmes pour les études empiriques : l'approche quantitative et l'approche qualitative :

Qualitative : ce paradigme étudie l'objet dans son environnement naturel. Il tente d'interpréter un phénomène sur base d'explications fournies par les sujets. Cette approche commence par l'acceptation qu'il existe plusieurs interprétations.

Quantitatif : ce paradigme tente de quantifier une relation ou comparer 2 groupes (ou plus) avec comme but d'identifier une relation de cause à effet. Il peut s'agir d'une expérience contrôlée ou d'une étude de cas. L'avantage est que la quantification permet la comparaison et l'utilisation de techniques statistiques.

En règle générale, les études quantitatives (*e.g.*, une expérience contrôlée) sont utilisées afin de déterminer les effets d'un traitement tandis que les études qualitatives sont utilisées afin d'expliquer les résultats d'une étude quantitative. Les 2 approches sont donc complémentaires et non compétitives.

Wohlin *et al.* [WRH+00] identifient dans la littérature 3 grands types d'études empiriques. Ils dépendent du but de l'étude, des conditions de l'étude et de la portée de l'étude (une technique, une méthode ou un outil).

Le sondage

C'est souvent une étude réalisée rétrospectivement (*e.g.*, lorsqu'une technique est déjà utilisée depuis un moment). Les données qualitatives ou quantitatives sont généralement récoltées par l'intermédiaire d'un questionnaire ou d'une entrevue. L'échantillon doit évidemment être représentatif de la population étudiée. Les résultats sont ensuite analysés afin de tirer des conclusions descriptives ou des explications.

Les sondages sont fort utilisés par les sciences sociales où, contrairement aux 2 autres types, il est impossible de manipuler les variables. Généralement, le but d'un sondage est la généralisation des résultats à la population étudiée et non pas au seul échantillon. Enfin, les sondages permettent

l'étude d'un grand nombre de variables même s'il est conseillé, afin de limiter la quantité de travail d'analyse, d'obtenir la plus grande compréhension à partir du plus petit nombre de variables.

Le but d'un sondage peut être :

Descriptif afin de valider une hypothèse concernant une population. L'intérêt ne porte pas sur la cause du phénomène, mais sur le phénomène lui-même.

Explicatif afin de faire valoir une explication à un phénomène observé. On peut par exemple étudier les relations entre différentes techniques et plusieurs variables pour expliquer pourquoi certains développeurs préfèrent une technique à une autre.

Exploratif afin de s'assurer qu'il n'y a pas de trop gros problèmes en vue avant une étude plus approfondie. Le but n'est donc pas de répondre à une question de recherche mais de fournir des résultats permettant d'améliorer l'investigation réelle future.

L'étude de cas

Les études de cas sont utilisées pour le suivi de projets, activités ou missions. Durant l'étude, les données sont récoltées dans un but particulier. Des analyses statistiques sont ensuite menées. Généralement, le but est de suivre un attribut particulier ou d'établir des relations entre différents attributs. Il s'agit d'une étude observatrice contrairement à l'expérimentation qui est une étude contrôlée.

Une étude de cas s'étend donc sur une période donnée, pendant laquelle on n'étudie qu'une seule entité/phénomène. Une multitude de moyens de récolte de données sont disponibles et dépendent de différents facteurs tels que l'échelle de l'évaluation. Ces études sont bien adaptées à l'évaluation des méthodes industrielles car elles permettent d'éviter ce problème d'échelle. Elles permettent de capter des changements distribués ou qui mettent du temps à se produire. Elles ont l'avantage d'être facile à planifier mais leur interprétation et généralisation sont difficiles.

L'étude de cas est une méthode standard des sciences telles que la sociologie, la médecine et la psychologie. En génie logiciel, elles peuvent être utilisées pour évaluer le comment et le pourquoi d'un phénomène mais peuvent également être utilisées afin de comparer 2 méthodes, outils, etc. Bien entendu, pour qu'une étude de cas soit valide, il est nécessaire que certains aspects fassent l'objet d'une attention particulière. Il faut par exemple tenter de minimiser l'impact des *facteurs confondants*, qui rendent impossible la distinction des effets d'un facteur par rapport à un autre.

L'expérimentation

Une expérimentation est généralement conduite dans un laboratoire, ce qui fournit un haut niveau de contrôle. Le but est de manipuler une ou plusieurs variables en fixant celles qui n'intéressent pas le chercheur. L'effet des manipulations de variables est alors mesuré et des analyses statistiques sont menées. On distingue la notion d'expérience de celle de quasi-expérience. La *quasi-expérience* est le cas où il est impossible d'effectuer une affectation aléatoire des sujets aux différents traitements. Nous ne parlerons par la suite que d'expérience et non plus de quasi-expérience, mais tout ce qui est dit vaut aussi pour les quasi-expériences.

Une expérimentation est une investigation formelle, rigoureuse et « contrôlée ». Les facteurs clés sont manipulés directement, précisément et systématiquement. Une expérimentation implique que les résultats de plusieurs traitements soient comparés. On distingue 2 types d'expérimentation :

Hors ligne : l'expérience est menée dans un laboratoire où les conditions sont contrôlées et où les événements sont organisés afin de simuler les conditions d'apparition du monde réel des phénomènes.

En ligne : l'expérience est menée sur le terrain dans les conditions normales d'apparition des phénomènes. Dans ce cas, le niveau de contrôle est moindre, mais non totalement absent.

Les expérimentations sont utiles afin d'étudier différents aspects tels que :
- Confirmer/tester des théories existantes.
- Confirmer des convictions conventionnelles.
- Explorer les relations de causalité entre variables.
- Évaluer l'exactitude d'un modèle.
- Valider des mesures.

Le processus sous-jacent à une expérimentation est présenté dans la section 3.2.

Comparaison des stratégies

Les pré-requis concernant une étude limite généralement le choix concernant la stratégie à adopter. Les différents facteurs à prendre en compte sont : le contrôle que le chercheur a sur l'étude, le degré de liberté que le chercheur a concernant les mesures à collecter, le coût de l'étude et la facilité avec laquelle une réplication de l'étude peut être réalisée. Le tableau 3.1 présente la comparaison des stratégies.

Le but d'une *réplication* est de valider les résultats d'une étude antérieure. On parlera d'une « vraie » réplication s'il est possible de reproduire la conception *et* les résultats d'une étude.

Facteur	Sondage	Étude de cas	Expérimentation
Contrôle de l'exécution	Non	Non	Oui
Contrôle des mesures	Non	Oui	Oui
Coût	Faible	Moyen	Élevé
Facilité de réplication	Élevée	Faible	Élevée
Stratégie qualitative/quantitative	Les 2	Les 2	Quantitative uniquement

TAB. 3.1: Comparaison des stratégies empiriques.

Environnement de recherche

Lorsqu'une organisation veut évaluer l'impact d'un changement dans un processus, les études empiriques permettent d'obtenir des informations quantifiées. Les 3 stratégies vues dans la section 3.1.3 remplissent alors chacune un rôle différent.

La figure 3.2 donne un exemple typique et place les différentes stratégies dans un environnement de recherche typique. Cette place est fonction de la taille « normale » des études pour chacune des stratégies. Cela signifie qu'il s'agit là de l'environnement généralement utilisé pour les différentes stratégies lorsque l'on désire mettre en pratique des résultats de recherche, mais qu'il n'est pas obligatoire de suivre ce placement.

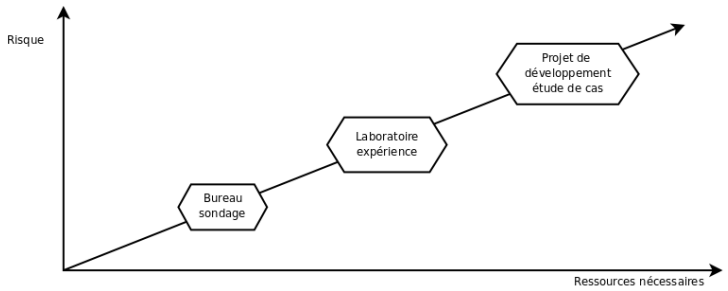


FIG. 3.2: Environnements de recherche (adpté de [WRH+00]).

Les environnements de recherche sont :

Bureau : la proposition de changement est évaluée hors-ligne, sans sa réalisation.

Laboratoire : la proposition de changement est évaluée dans un laboratoire où une expérimentation est menée sur une partie limitée du processus et exécutée sous contrôle.

Projet de développement : la proposition de changement est évaluée dans une situation de développement réel/en ligne (*e.g*, lors d'un projet pilote).

La figure 3.2 montre bien l'augmentation progressive des ressources nécessaires et des risques généralement admise par l'industrie. Cela ne signifie pas qu'une stratégie est moins coûteuse qu'une autre. Le coût dépend du type et de la taille de l'étude ; à différentes échelles correspondent mieux différentes stratégies. Il est bien entendu préférable de commencer avec peu de risques avant de se lancer dans une grosse étude de cas.

Sjøberg *et al.* [Sjo07] font également la différence entre étude primaire et étude secondaire :

Primaire : de telles recherches impliquent la récolte et l'analyse de données brutes en utilisant notamment les stratégies discutées dans la section 3.1.3 (sondage, étude de cas et expérimentation).

Secondaire : de telles études utilisent les données existantes précédemment publiées dans un but de synthèse : résumer une même famille de méthodes, intégrer et lorsque c'est possible combiner les résultats. Ce type d'études peut également identifier des champs importants qui n'ont pas encore fait l'objet d'études empiriques. Ce type d'études est né avec le constat que, quelle que soit la justesse de la conception et de la conduite de chacune des expérimentations, leurs résultats ne s'appliquent qu'à leurs niveaux de généralisation respectifs. Le but est donc de généraliser encore par recoupement. Un bon exemple de ce type d'études est celle de Duchowski intitulée : « Eye Tracking Methodology : Theory and Practice » [Duc07].

3.1.4 Mesures

Les mesures sont primordiales lorsque l'on désire avoir le contrôle d'un projet, d'un produit ou d'un processus. Pour contrôler une étude et identifier les effets, nous devons être capables de mesurer les entrées et les sorties afin de pouvoir décrire les relations de cause à effet. Les mesures sont considérées comme étant, par définition, la correspondance entre le monde empirique et le monde numérique [HALS08]. Au lieu de faire son jugement directement sur l'objet réel, nous étudions les mesures et faisons notre jugement dessus [WRH$^+$00].

Une des caractéristiques de base des mesures, est qu'elles doivent préserver des propriétés de l'observation de l'attribut considéré. Si l'objet A est plus long que l'objet B, la mesure de A doit être plus grande que la mesure de B. Wohlin*et al.* [WRH$^+$00] définissent donc le concept de mesure valide. Une *mesure valide* doit respecter les caractéristiques suivantes :

- Elle doit être une caractérisation mathématique de l'attribut.
- Elle doit permettre la distinction entre 2 objets différents mais, dans la limite de l'erreur de mesure, des objets doivent pouvoir avoir la même valeur.
- Elle doit préserver notre notion intuitive de l'attribut et la manière avec laquelle nous différencions différents objets.

La transformation d'un attribut en une mesure peut se faire selon différentes familles d'échelles :

Nominale : c'est l'échelle la moins puissante. Elle fait correspondre à l'attribut un nom ou un symbole. Les noms ou les symboles ne présentent aucune relation d'ordre.

Ordinale : cette échelle ordonne les entités selon un critère. Il y a une relation d'ordre.

Intervalle : avec ce type d'échelle, en plus de la relation d'ordre, la notion de différence entre 2 mesures a un sens mais pas la mesure en elle-même.

Rapport : avec ce type d'échelle, en plus de la relation d'ordre et de la notion de différence, la notion de rapport a un sens. Il existe donc une valeur 0.

Il est dès lors possible de vouloir transformer une mesure définie sur une échelle vers une mesure définie sur une autre échelle. D'une manière générale, on ne peut effectuer de transformation que

vers une échelle au moins aussi puissante que l'échelle de départ. Une *transformation admissible* est une transformation qui préserve les relations entre objets et ne peut donc se faire qu'au sein d'une même famille d'échelles. Les études qualitatives se cantonnent généralement à l'utilisation d'échelles nominales et ordinales.

Il arrive que l'on ne puisse mesurer un attribut sans prendre en compte le point de vue de celui qui prend la mesure. En fonction de ce point de vue, les mesures sont divisées en 2 classes :

Mesure objective : c'est une mesure pour laquelle il n'y a aucun jugement dans la valeur attribuée. Elle ne dépend donc que de l'objet mesuré. Une prise de mesure répétée fournira inévitablement le même résultat.

Mesure subjective : c'est l'opposé. La personne effectuant la mesure contribue par un jugement quelconque. La mesure dépend alors de l'objet et du point de vue adopté par celui prenant la mesure. Il s'agit en général de mesures définies sur une échelle nominale ou ordinale.

Il arrive aussi que l'attribut qui nous intéresse ne soit pas mesurable directement. La mesure est alors dérivée d'autres mesures qui sont, elles, directement mesurables. On parle alors de mesure directe ou de mesure indirecte.

Habra*et al.* [HALS08] soulignent qu'une des particularités du génie logiciel est son manque de maturité et que cela se fait cruellement sentir par le manque de consensus autour du problème des mesures. Une discipline mature dispose d'un éventail d'instruments de mesure reconnus internationalement, ce qui n'est pas encore le cas du génie logiciel pour lequel ils identifient au sein de l'ISO pas moins de 4 méthodes de mesure reconnues. C'est pourquoi ces auteurs proposent un précieux glossaire relatif aux méthodes de mesure intégrant les problèmes théoriques et pratiques de la littérature et de l'industrie. Ils fournissent également un cadre de référence proposant une structure d'analyse pour la compréhension et l'évaluation des méthodes de mesure existantes.

Le problème de définition des mesures est très important en oculométrie et est traité plus en détail dans la section 6.2.4.

3.2 La stratégie expérimentale

Le point de départ de toute expérience est une idée de relation de cause à effet éventuelle qu'il est possible de formuler selon une hypothèse formelle. Le but de l'expérience est alors de vérifier cette hypothèse. L'expérience permet de contrôler les éléments qui entrent en jeu. Une fois l'expérience réalisée, la relation entre les traitements et les résultats est analysée. Dans le cas où l'expérience est correctement conçue, il est possible de tirer des conclusions concernant la relation de cause à effet pour laquelle a été formulée l'hypothèse. Ce principe général est illustré par la figure 3.3. Les numéros font référence aux menaces pesant sur une expérience. Ces menaces sont abordées à la fin de la section 3.2.4.

La réalisation d'une expérience est complexe. Cette complexité justifie le besoin d'un processus. Le but de cette section est de présenter ce processus. Nous commençons par formaliser, dans la section 3.2.1, certaines notions déjà utilisées précédemment avant de passer en revue, dans les sections 3.2.2 à 3.2.7, les différentes étapes nécessaires à la réalisation d'une expérience. A l'instar de la section précédente, cette section est inspirée par l'ouvrage de Wohlin *et al.* « Experimentation in software engineering : An Introduction » [WRH+00]. Elle est également complétée de certaines notions issues de l'ouvrage « Statistics in a Nutshell » [BW08].

3.2.1 Vocabulaire

Deux types de variables entrent en jeu lors de la réalisation d'une expérience : les *variables dépendantes* et les *variables indépendantes* :

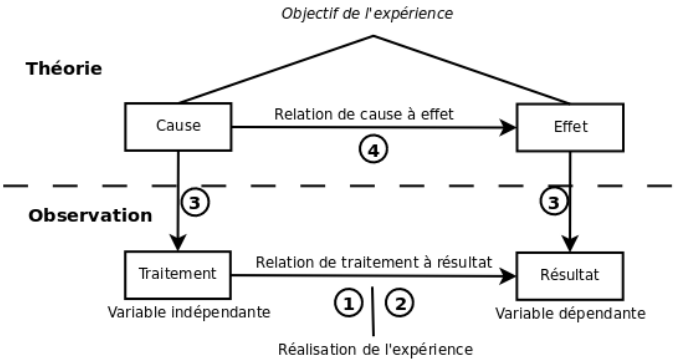


FIG. 3.3: Principe d'une expérience et de sa validité (adapté de [WRH+00]).

Variables indépendantes : ce sont les variables soit manipulées, soit fixées par l'expérimentateur.

Variables dépendantes : ce sont les variables pour lesquelles l'expérimentateur désire étudier le changement provoqué par les manipulations des variables indépendantes. L'expérimentateur s'attend à ce que leurs valeurs soient influencées par celles des variables indépendantes.

La valeur de certaines variables indépendantes est fixée par l'expérimentateur. Cela permet d'étudier l'effet du changement de valeur des variables indépendantes non fixées. Ces variables indépendantes non fixées sont appelées *facteurs*. On définit aussi la notion de *traitement* qui est le nom donné à une valeur particulière d'un facteur. Le choix d'un traitement et le niveau auquel les autres variables indépendantes sont fixées font partie de la conception d'une expérience. Ces notions sont illustrées par la figure 3.4. On définit aussi la notion de facteur confondant qui est une variable indépendante de confusion capable d'introduire un biais dans l'analyse des liens entre variables.

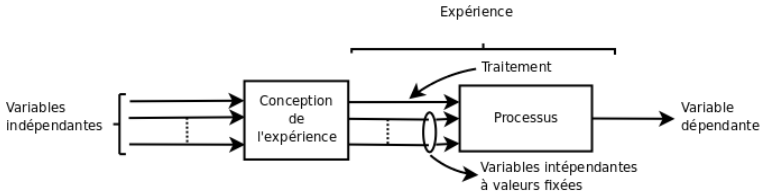


FIG. 3.4: Illustration d'une expérience (adapté de [WRH+00]).

Les traitements sont appliqués à une combinaison d'objets et de sujets. Un *sujet* est une personne prenant part à l'expérience. Un *objet* est le support sur lequel le sujet exécute sa tâche. Enfin, une expérience est constituée d'un ensemble de tests. Chaque test est la combinaison d'un traitement, d'un sujet et d'un objet.

3.2.2 Processus d'une expérience

Les processus décrivent les activités nécessaires à la réalisation d'une tâche d'une certaine ampleur. Ils sont importants car ils peuvent servir de « checklist » et de lignes conductrices. Le point de départ de toute expérience est une idée ainsi que la conviction qu'une expérience permet son évaluation. Il est important de préparer une expérience minutieusement. L'ensemble du processus présenté est illustré par la figure 3.5.

Ce processus ne suit pas le modèle en cascade. Il n'est pas obligatoire qu'une activité soit entièrement terminée pour que la suivante débute. Le processus est légèrement itératif, il peut être utile

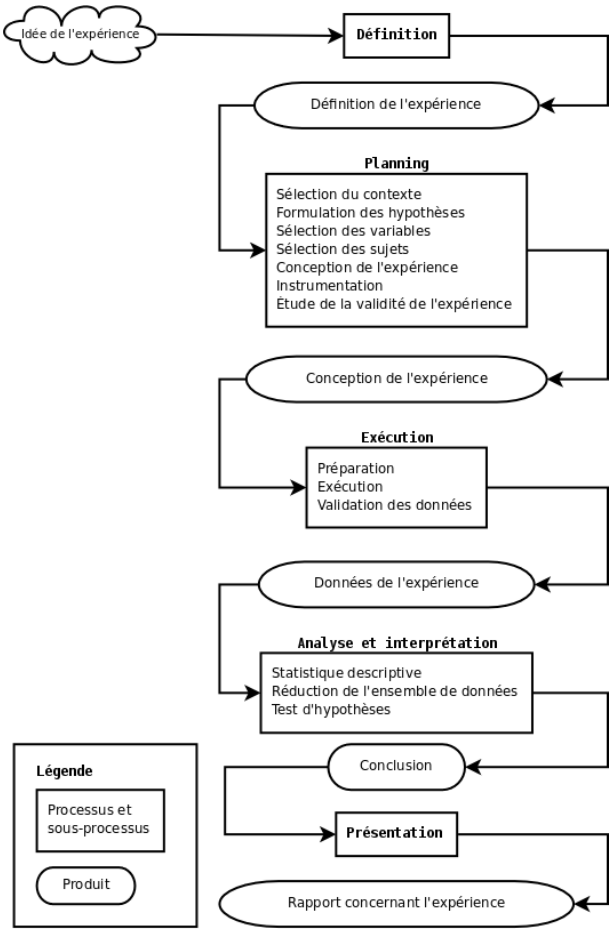


FIG. 3.5: Processus d'une expérience (adapté de [WRH+00]).

de revenir en arrière afin de raffiner avant de continuer. La seule exception se situe au niveau de l'exécution de l'expérience. Une fois que cette activité a commencé, il est interdit de revenir sur les activités de définition et planification, sous peine d'influencer les résultats. Il deviendrait alors impossible d'utiliser les mêmes sujets de retour à l'activité d'exécution. Sachant que les sujets en génie logiciel ne sont pas faciles à recruter, il vaut mieux respecter cette exception.

3.2.3 Définition de l'expérience

Le but de l'*activité de définition* est de poser clairement les objectifs de l'expérience. Il faut en déterminer les fondations. De mauvaises fondations peuvent faire échouer l'expérience. En effet, le paradigme utilisé est le paradigme But/Question/Mesure (GQM : Goal/Question/Metric). Cette approche se base sur le fait qu'il faut en premier lieu spécifier le but d'un projet. Ensuite, on lie ce but aux données, c'est-à-dire définir le but d'une manière opérationnelle. Quelle question doit-on se poser afin d'atteindre notre but. Enfin, on définit un cadre d'interprétation des données en rapport avec le but en choisisant ou définissant les mesures que l'on va utiliser.

WOHLIN *et al.* tirent de la littérature un cadre de référence qui permet de s'assurer de la bonne définition de l'expérience. Ce cadre se présente comme ceci :

> Analyse du (des) <Objet(s) d'étude>
> dans le but de <Objectif >

selon la <Focalisation sur la qualité>
sous l'angle de <Perspective>
dans le contexte du <Contexte>

Objet(s) d'étude : c'est l'entité étudiée par l'expérience.

Objectif : c'est la définition de l'intention de l'expérimentateur.

Focalisation sur la qualité : c'est l'effet primaire analysé par l'expérience.

Perspective : c'est le point de vue adopté lors de l'interprétation des résultats.

Contexte : c'est l'environnement dans lequel se déroule l'expérience : sujets et objets.

3.2.4 Planification de l'expérience

Après avoir défini le « pourquoi » de l'expérience, il faut définir le « comment » avec la *planification*. Une expérience à la planification bâclée ne remplira pas ses objectifs. Sur base de la définition fournie par l'activité précédente, il est possible d'aborder les 6 sous-activités de la planification : la *sélection du contexte*, la *formulation des hypothèses*, la *sélection des variables*, la *sélection des sujets*, la *conception* de l'expérience, l'*intrumentation* et enfin l'*évaluation de la validité* de l'expérience.

Sélection du contexte

Le contexte d'une expérience peut être caractérisé par 4 dimensions :
- Hors ligne ou en ligne.
- Étudiant ou professionnel.
- Problèmes de tailles limitées ou problèmes réels.
- Spécifique ou général.

Réaliser une expérience sur un projet réel (*en ligne*) impliquant des professionnels et un produit réel comporte des risques pour le produit (*e.g.*, la méthode testée n'est pas aussi efficace que prévu). Il est plus sage de réaliser l'expérience en parallèle avec les autres projets *hors ligne*. Les risques sont réduits mais les coûts augmentent. Une solution moins coûteuse est de réaliser l'expérience sur des projets *étudiants* plutôt que *professionnels*. Cette solution est moins coûteuse mais les résultats ne s'appliquent alors *a priori* pas à l'industrie. Enfin, les coûts et le temps imparti font que ces projets ne sont généralement pas de *réels projets* mais des *projets de taille limitée*. La validité des résultats de l'expérience peut donc entrer dans un intervalle borné par un contexte hyper-spécifique et le contexte général du génie logiciel.

Perry *et al.* [PPV00] soulignent qu'il est souvent nécessaire de réaliser plusieurs expériences pour répondre à une problématique et qu'il faut donc trouver un moyen pour obtenir l'information à moindre coût. Cela implique le traitement de problèmes réduits ou plus précis.

Formulation des hypothèses

L'analyse statistique la plus souvent utilisée pour une expérience est le test d'hypothèse. Ce sujet est traité plus en profondeur par les sections 4.1 et 4.2, nous nous contentons donc de n'en donner ici qu'un bref aperçu.

Le principe est de poser une hypothèse que l'on va tenter de rejeter au profit d'une hypothèse alternative qui est celle que l'on pense réelle. Imaginons que l'on ne sache pas encore que la taille d'un être humain varie avec son âge mais que l'on soit persuadé que c'est le cas. L'hypothèse à tester serait du type « La taille d'un homme ne varie pas avec le temps » et l'hypothèse alternative

serait « La taille d'un homme varie avec le temps ». L'expérience tentera de montrer que la première hypothèse est fausse, ce qui confirmerait la seconde hypothèse. Le cas où la première hypothèse n'est pas rejetée ne la rend pas pour autant vérifiée.

Il faut donc formuler au moins 2 hypothèses :

L'hypothèse nulle (H_0) : cette hypothèse statue sur le fait que les différences observées sont le fruit du hasard et qu'il n'y a pas de relation de cause à effet. Elle statue sur l'égalité entre 2 phénomènes (*e.g.*, l'égalité de 2 moyennes).

L'hypothèse alternative (H_1) : il s'agit de l'inverse de l'hypothèse nulle. Il peut y avoir plusieurs hypothèses alternatives. Dans le cas où l'hypothèse nulle statue sur l'égalité entre 2 moyennes $\mu_1 = \mu_2$, les hypothèses alternatives peuvent être : $\mu_1 < \mu_2$, $\mu_1 > \mu_2$ ou $\mu_1 \neq \mu_2$.

Cette structure a pour conséquence que l'on ne peut pas tester tout ce que l'on veut. Il faut arriver à formuler, et tester, l'inverse de l'hypothèse que l'on désire vérifier. Le type d'hypothèse que l'on peut formuler dépend des tests statistiques d'hypothèses disponibles.

Sélection des variables

La sélection des variables est une partie importante de la planification. Le choix des variables indépendantes n'est pas simple et requiert bien souvent une certaine connaissance du domaine de recherche. Elles doivent avoir un effet sur les variables dépendantes et être contrôlables. Le choix des variables dépendantes se fait lui sur base des hypothèses. Ces variables permettent de mesurer l'effet des traitements. Elles sont souvent des mesures indirectes qui doivent être soigneusement étudiées car elles affectent directement le résultat. Il n'y en a généralement qu'une mais afin d'augmenter la validité de l'expérience, il peut être utile d'en définir plusieurs pour recouper les résultats. D'une manière générale, le choix d'une variable s'accompagne du choix du type d'échelle utilisé. Après cette sélection, les hypothèses sont généralement raffinées.

Il y a 3 types de variables dépendantes : les *références*, les *réponses* et les *intermédiaires* [BW08] :

Variable de référence : elles permettent de mesurer les différences induites par un traitement censé modifier le comportement des sujets, par rapport à un traitement de base censé représenter le comportement normal des sujets. Dans notre expérience, ce type de variable est utilisé afin de mesurer la différence induite par l'introduction d'un patron de conception.

Variable de réponse : ce sont les variables dépendantes « de base ». Elles permettent de vérifier l'existence d'une relation de cause à effet.

Variable intermédiaire : elles sont utilisées afin d'expliquer une relation indirecte mais contrôlable entre le traitement et la variable de réponse. Selon certaines conceptions, la distinction entre traitement et variable intermédiaire n'a pas d'importance. Un chimiste intéressé par les propriétés chimiques de l'eau peut se contenter de travailler au niveau atomique plutôt que de travailler au niveau sub-atomique. Or, les propriétés observées au niveau atomique dépendent des propriétés du niveau sub-atomique.

Il est toujours intéressant de donner aux variables des noms ayant un rapport avec leur rôle. Enfin, selon Boslaugh et Watters [BW08], certains auteurs voudraient inverser les dénominations de variables dépendantes et indépendantes. Les variables dépendantes sont alors celles que le chercheur est en mesure de contrôler (elles dépendent de lui) tandis que les variables indépendantes, qu'il ne peut contrôler, sont les variables de réponse.

Une discussion concernant les variables dépendantes spécifiques au domaine de l'oculométrie est présentée dans la section 6.2.4.

Sélection des sujets

La *population* est l'ensemble des personnes que le statisticien étudierait s'il en avait la possibilité. C'est donc le groupe de personnes auquel le statisticien aimerait généraliser ses résultats. Mais

presque toutes les études sont basées sur l'étude d'un échantillon tiré de la population plutôt que de la population entière. Il n'existe que quelques exceptions (*e.g.*, l'enquête socio-économique menée en 2001 par l'Institut national de statistique Belge [NdS01]).

La sélection de l'échantillon étudié est d'une grande importance car, à ce stade, un biais irrévocable est susceptible de s'introduire. Un échantillon correct doit être sélectionné de manière aléatoire. Or, il n'est pas évident dans la réalité d'obtenir une telle sélection [2]. Afin de s'assurer de la validité des résultats de l'expérience, il vaut mieux se limiter à une population particulière.

Il existe 2 types d'*échantillonnage* : probabiliste et non probabiliste. Idéalement, l'échantillonnage doit être probabiliste, malheureusement, certaines des méthodes les plus utilisées ne le sont pas.

Les méthodes probabilistes sont par exemple :

Échantillonnage simplement aléatoire : les sujets sont sélectionnés aléatoirement parmi la population.

Échantillonnage systématique : le premier sujet est sélectionné aléatoirement dans la liste de la population. Le sujet suivant est la n^{e} personne suivant le sujet précédemment sélectionné et ainsi de suite.

Échantillonnage aléatoire en strates : la population est divisée en groupes avec une distribution connue. Un échantillonnage aléatoire est ensuite opéré sur chacun des groupes.

Les méthodes non probabilistes sont par exemple :

Échantillonnage de convenance : la personne qui convient le plus est sélectionnée.

Échantillonnage volontaire : l'échantillon est composé de volontaires.

La taille de l'échantillon a également une importance sur la généralisation possible. Plus l'échantillon est large, plus la puissance de l'expérience le sera (voir section 4.1.4). C'est encore plus important dans le cas où il y a une forte variabilité au sein de la population.

Conception de l'expérience

Les analyses statistiques applicables dépendent de la conception de l'expérience. La conception et l'interprétation des résultats sont donc liées. C'est pourquoi, il est nécessaire de bien définir le problème que l'on veut étudier. Une définition claire (voir section 3.2.3) est une bonne aide à la compréhension et à l'élaboration de la solution.

La série de tests de l'expérience doit être soigneusement préparée. Il faut commencer par analyser les hypothèses afin de voir quel test statistique permettra de rejeter l'hypothèse nulle. La conception de l'expérience est alors faite sur base des conditions nécessaires à la réalisation du test statistique sélectionné. On décide également du nombre de tests qu'il faut effectuer afin de faire ressortir l'effet du traitement.

Il existe 3 principes généraux concernant la conception d'une expérience :

Randomisation : tous les tests statistiques nécessitent que les données soient observées à partir de variables indépendantes aléatoires. La randomisation s'applique à l'allocation des objets, des sujets et à l'ordre dans lequel les tests sont présentés. La randomisation est utilisée pour atténuer les effets des facteurs présents autres que le traitement.

Blocage : ce principe consiste à éliminer l'effet d'un facteur qui n'intéresse pas le chercheur mais qui influence la réponse. Ce n'est possible que si le facteur est contrôlable. Le blocage est systématiquement utilisé pour éliminer les effets non désirés dans la comparaison de traitements.

Équilibrage : affecter les traitements afin que chacun ait un même nombre de sujets donne une conception équilibrée. Cela simplifie l'analyse des données et la rend plus robuste. Ce principe n'est cependant pas indispensable.

[2] Certains docteurs en statistique sont uniquement spécialisés dans ce domaine.

Une expérience peut faire intervenir un ou plusieurs facteurs comme elle peut faire intervenir un ou plusieurs traitements. Des exemples de types de conception sont :
- 1 facteur avec 2 traitements.
- 1 facteur avec plus de 2 traitements.
- 2 facteurs avec 2 traitements.
- Plus de 2 facteurs, chacun avec 2 traitements.

Instrumentation

Il y a 3 types d'instruments pour une expérience : les objets, les lignes de conduite et les instruments de mesure. Durant la phase de planification, les instruments sont choisis et développés pour les besoins spécifiques de l'expérience avant l'exécution de celle-ci. Le but de la phase d'instrumentation est de se donner les moyens nécessaires à l'exécution de l'expérience sans affecter le contrôle qu'on en a. Idéalement, les résultats doivent être les mêmes quelle que soit l'instrumentation. Si l'instrumentation influence les résultats, alors, l'expérience n'est pas valide.

Analyse de la validité

La validité des résultats d'une expérience est évidemment un point important. C'est pourquoi, il est important de considérer cette question dans la phase de planification. Une première chose est d'obtenir des résultats valides pour la population source de l'échantillon sélectionné. Ensuite, il peut être intéressant de pouvoir généraliser les résultats à une population plus large. Wohlin *et al.* [WRH+00] tirent de la littérature 4 types de menaces à la validité d'une expérience. Ce sont les menaces à la conclusion, internes, de construction et externes.

La figure 3.3 met en relation ces menaces avec le principe d'une expérience. Pour tirer une conclusion, il y a 4 étapes à franchir et chacune d'elles correspond à un type de validité :

1. **Validité de conclusion :** elle concerne la relation entre le traitement et le résultat. Il faut s'assurer qu'il y a une relation statistique.
2. **Validité interne :** si l'on observe une relation entre le traitement et le résultat, il faut être sûr que c'est bien le traitement qui en est la raison.
3. **Validité de construction :** elle concerne la relation entre la théorie et les observations. Si nous sommes bien dans une relation de cause à effet, il faut s'assurer que :
   - le traitement reflète bien la cause (partie gauche de la figure 3.3).
   - le résultat reflète bien l'effet (partie droite de la figure 3.3).
4. **Validité externe :** elle concerne le pouvoir de généralisation de l'expérience. Est-ce que la relation de cause à effet observée dans l'expérience peut être généralisée en dehors du cadre de celle-ci ?

A chacune des validités présentées correspondent des menaces pouvant survenir. Wohlin *et al.* [WRH+00] proposent une liste pratiquement exhaustive des menaces pesant sur une expérience en génie logiciel. Cette liste est utilisée pour l'analyse de la validité de notre expérience dans le chapitre 9. Il est important de noter que **l'expérience parfaite n'existe pas**. C'est la conséquence des différents types de validités qui peuvent se montrer exclusifs. Ainsi, renforcer un type de validité peut nuire à un autre. L'expérimentateur doit établir une priorité dans les menaces à mitiger et en accepter certaines.

3.2.5 Exécution de l'expérience

C'est la phase où les traitements sont effectivement appliqués aux sujets. C'est la partie de l'expérience où l'expérimentateur rencontre les sujets. Même si une expérience a été parfaitement

conçue avec la bonne méthode d'analyse, les conclusions seront erronées si les sujets n'y ont pas participé sérieusement. Heureusement, les sciences sociales, impliquant pratiquement toujours le facteur humain pour leurs expériences, ont élaboré des lignes de conduite pouvant en grande partie être réutilisées par le génie logiciel. La phase d'exécution est divisée en 3 étapes : la préparation, l'exécution proprement dite et la validation des données.

Préparation

L'exécution proprement dite d'une expérience nécessite une préparation. Meilleure est la préparation de l'expérience, meilleure sera son exécution. Il y a généralement 2 points à prendre en compte : le recrutement et la préparation du matériel prévue par l'activité d'instrumentation.

Recrutement des participants. Sans sujets, l'expérience ne peut être menée à bien. Il est important que les sujets soient motivés. Il faut aussi que l'activité du sujet en dehors de l'expérience ressemble aux tâches qui lui seront demandées pour l'expérience. Plusieurs autres aspects doivent également être considérés :

- Le consentement : les participants doivent être d'accord avec les objectifs de l'expérience. D'abord pour une raison éthique évidente, ensuite parce que la compréhension des objectifs de l'expérience peut affecter leurs comportements, une mauvaise compréhension pourra dès lors affecter négativement les résultats de l'expérience. Il est également sage de les rassurer en leur faisant savoir qu'ils peuvent se retirer de l'expérience sans conséquence à tout moment.
- Sensibilité des résultats : un sujet peut considérer ses résultats comme des données sensibles. Il est dès lors judicieux (dans la mesure du possible) de rendre ces résultats confidentiels et de l'en informer.
- Incitation : un moyen d'inciter les personnes à prendre part à l'expérience est de leur donner un « incitatif ». Cet incitatif ne peut cependant pas être trop important car cela ne motiverait pas que des participants sérieux.
- Tromperie : tromper un participant est une mauvaise chose. Si la tromperie est inévitable, elle ne doit pas se faire à propos de données sensibles pour le sujet et doit être révélée et expliquée dès que possible.

Préparation du matériel. Les instruments nécessaires à l'exécution de l'expérience ont été définis antérieurement, il faut maintenant les préparer. Certaines données seront collectées auprès des sujets. Il s'agit souvent dans ce cas d'un formulaire à remplir. Il faut aussi déterminer si les données sont rendues anonymes. Dans la mesure du possible (*e.g.*, s'il n'y a pas d'autre(s) étude(s) prévue(s) avec le même sujet) les données doivent être rendues anonymes. Des traitements différents pouvant être appliqués à différents sujets, il est souvent utile de faire une préparation personnalisée. Lors d'entrevues, il peut également être intéressant de prévoir des questions différentes pour les sujets.

Exécution

Il y a plusieurs manières d'exécuter une expérience. Cela va des expériences exécutées via un grand rassemblement aux expériences exécutées sur plusieurs années. La première manière offre quelques avantages. L'expérience n'est expliquée qu'une fois et, après le rassemblement, l'expérimentateur dispose directement de l'ensemble des données. De plus, les problèmes peuvent être résolus sur le champ. Dans le second cas, il est impossible pour l'expérimentateur de contrôler tous les détails de la collecte des données, ce qui augmente le risque de la présence de facteurs confondants.

La collecte des données. La collecte peut se faire sous plusieurs formes : le remplissage manuel d'un formulaire, le remplissage partiellement automatisé d'un formulaire, une entrevue ou automatiquement par un outil. Chaque méthode a ses avantages et ses inconvénients. Le remplissage manuel ne nécessite que peu d'efforts de la part de l'expérimentateur mais l'empêche d'identifier directement les problèmes liés au formulaire. L'entrevue permet une meilleure communication avec le sujet durant la collecte mais demande aussi plus d'efforts.

L'environnement de l'expérience. Une expérience menée au sein d'un projet de développement régulier ne doit pas l'affecter plus que nécessaire. L'expérience a pour but d'observer les effets de différents traitements dans un environnement. Si l'environnement change à cause de l'expérience, ces effets sont perdus. Il peut y avoir quelques exceptions. Si l'expérience révèle que certaines parties du projet peuvent être effectuées de manière plus efficace, ou si certaines estimations sont fausses, il peut être intéressant que l'expérimentateur avertisse le chef du projet. Cela peut même motiver le personnel du projet à participer à l'expérience.

Validation des données

Une fois les données collectées, l'expérimentateur doit vérifier qu'elles sont raisonnables et qu'elles ont été collectées correctement. Cela va de la compréhension des formulaires par les participants à la suppression de certaines données jugées aberrantes (ce problème est discuté dans la section suivante). Il est important de s'assurer que l'expérience s'est déroulée de la manière prévue, autrement les données peuvent êtres invalides. Un moyen de s'assurer de la bonne compréhension de l'expérience par les sujets est de leur présenter les données collectées. Cela leur donne un moyen d'exprimer leurs désaccords éventuels.

3.2.6 Analyse et interprétation des données

L'interprétation de données est composée de 3 étapes. Les statistiques descriptives permettent de prendre connaissance des données. Si certaines d'entre elles sont aberrantes, il peut être judicieux d'effectuer une réduction de l'ensemble des données. Enfin, les tests d'hypothèse permettent d'évaluer statistiquement les hypothèses.

Statistique descriptive

La statistique descriptive permet de visualiser et présenter l'ensemble des données. Elle permet de se faire une première idée des données avant d'effectuer les tests d'hypothèse. En particulier, elle permet d'isoler certaines données aberrantes, appelées valeurs solitaires ou extrêmes.

Le type d'échelle utilisé pour les variables restreint les techniques statistiques utiles. Le tableau 3.2 présente certaines techniques utiles par rapport au type d'échelle. Les mesures appliquées à une échelle peuvent toujours s'appliquer à une échelle plus puissante. Ainsi, le mode peut s'appliquer à tous les types d'échelle.

Type d'échelle	Mesure de la tendance centrale	Mesure de la dispersion
Nominale	Mode	Fréquence
Ordinale	Médian, centile	Intervalle de variation
Intervalle	Moyenne	Variance, écart-type
Rapport	Moyenne géométrique	Coefficient de variation

TAB. 3.2: Types d'échelle et statistiques.

Les représentations graphiques permettent une visualisation plus intuitive des données. Ce sont les histogrammes, camemberts, etc. Une technique fort utilisée est la boîte à moustache. Cette

technique permet de visualiser plusieurs caractéristiques importantes de l'ensemble de données : la médiane, le 1er et 3e quartiles (respectivement le 25e et 75e centile), la tendance centrale, la symétrie et la présence de valeurs solitaires. De plus, 2 boîtes à moustache côte à côte permettent une comparaison rapide de 2 ensembles de données. Ce concept de boîte à moustache est illustré par la figure 3.6.

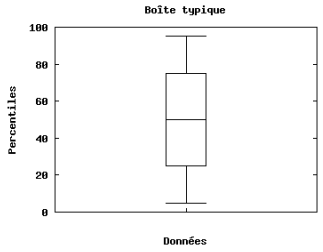


FIG. 3.6: Boîte à moustache typique.

La boîte en elle-même est construite en partant du 1er quartile pour aller jusqu'au 3e. Elle est coupée par la médiane. Les 2 extrémités qui donnent une forme de moustache à l'ensemble sont généralement tracées jusqu'aux 5e et 95e percentiles (ou 1er et 9e décile). Les valeurs hors de ces « moustaches » sont caractérisées de valeurs extrêmes.

Réduction de l'ensemble de données

Quelle que soit la technique utilisée pour analyser les données, elle dépend de la qualité des données fournies. Si les données ne représentent pas ce que l'on pense, les conclusions tirées à partir de celles-ci seront erronées. Les erreurs peuvent venir de valeurs solitaires. Une *valeur solitaire* est une valeur beaucoup plus grande ou petite que l'ensemble des autres. Certaines représentations graphiques (*e.g.*, diagramme de dispersion) permettent une rapide et facile détection de telles valeurs.

Une fois la ou les valeurs solitaires détectées, l'expérimentateur doit décider de ce qu'il en fait. Il est important d'analyser les raisons d'une telle valeur. Si elle est due à un événement qui ne se produira plus (*e.g.*, le sujet a mal compris la tâche), elle devrait être éliminée. Si par contre, la valeur est due à un événement rare mais qui peut se reproduire (*e.g.*, la performance d'un sujet inexpérimenté), la valeur devrait être conservée car elle est instructive. Si la valeur est due à une variable non considérée jusqu'ici, il peut être intéressant d'étudier le phénomène sur base de cette variable. On peut par exemple étudier le phénomène avec du personnel ayant une expérience normale (en supprimant la valeur) et avec le personnel à faible expérience inclus (en ne supprimant pas la valeur).

Test d'hypothèse

Les tests d'hypothèse ont déjà brièvement été abordés dans la section 3.2.3 et le seront encore dans la section 4.1.3. Nous nous contenterons donc d'en rappeler la base. Le but d'un test d'hypothèse est de réfuter, si possible, une hypothèse nulle (H_0) sur base d'échantillons des populations étudiées. L'hypothèse nulle doit donc être l'inverse de l'intention de l'expérience. Si l'hypothèse nulle n'est pas rejetée, aucune conclusion ne peut être formulée. Si l'hypothèse nulle est rejetée, c'est au profit d'une hypothèse alternative qui est une formulation précise et formelle de ce que l'expérimentateur veut montrer.

Concrètement, il existe 2 types de tests d'hypothèse : les tests paramétriques et les tests non paramétriques :

Tests paramétriques : ces tests sont basés sur un modèle impliquant que les données respectent certaines contraintes (*e.g.*, être une approximation de la loi noramle). Ils impliquent également que les paramètres soient mesurés sur au moins une échelle de type d'intervalle.

Tests non-paramétriques : basés sur des présuppositions moins fortes, si ce n'est aucune, ces tests sont plus généraux que les tests paramétriques. Ils peuvent donc être utilisés à la place de ceux-ci mais non l'inverse. Ils sont néanmoins plus faibles.

Le choix d'un test est donc soumis à 2 conditions : l'applicabilité et la puissance du test. Les tests paramétriques sont plus puissants mais moins souvent applicables. Cependant, selon Brian *et al.* [BEM+96], il peut, **dans certains cas**, être utile d'utiliser les méthodes paramétriques même lorsque certaines conditions ne sont pas respectées.

Conclure l'analyse

Une fois les données analysées, il faut tirer les enseignements. Si l'hypothèse nulle a pu être rejetée, on peut envisager une relation de cause à effet entre les variables indépendantes et dépendantes. D'un autre côté, ne pas avoir rejeté l'hypothèse nulle signifie simplement qu'il n'y a pas de différence statistiquement significative entre les traitements. Si une différence statistiquement significative est trouvée, il faut considérer la validité externe de l'expérience avant de généraliser les résultats.

Une différence fortement statistiquement significative n'implique pas toujours une portée pratique. Une étude statistiquement significative montrant qu'une nouvelle technique diminue les coûts de 0.01% n'a pas vraiment de portée pratique. Par contre, une étude statistiquement moins significative montrant que la différence de coût est de 10% peut avoir une portée pratique importante. Cet « effet de taille » est à garder en mémoire. De plus, que l'hypothèse nulle n'ait pas été rejetée ne veut pas dire qu'elle est vraie. Il peut y avoir eu un problème dans la conception de l'expérience. Il faudrait peut-être refaire l'expérience en apportant plus de soin à sa préparation. Enfin, le fait de trouver une relation de cause à effet statistiquement significative **n'est pas une preuve** que la relation existe. Un troisième facteur non considéré peut en être la cause.

3.2.7 Présentation de l'expérience

Une fois l'expérience terminée, il est important de bien en présenter les résultats. Cette activité est importante pour être sûr que les leçons apprises soient prises en compte de la bonne manière. De plus, comme une expérience ne donne jamais de réponse définitive à une question, il faut faciliter les réplications éventuelles. Il faut donc fournir toutes les informations nécessaires.

Chapitre 4

Fondements mathématiques

Une expérience, que ce soit en génie logiciel ou autre, se base sur la statistique inférentielle et ses tests d'hypothèse afin d'analyser les données récoltées. Ce chapitre expose les principes basiques liés à la statistique inférentielle dans la section 4.1. Notre expérience utilise comme test d'hypothèse le 2-way Anova qui est un test d'Anova factoriel. La section 4.2 présente les éléments nécessaires à la compréhension de l'interprétation des résultats de ces tests d'Anova factoriels. Enfin, la section 4.3 présente quelques éléments techniques nécessaires à la compréhension du processus de classification des sujets en experts ou novices, présenté dans l'analyse des données (section 8.3).

4.1 Statistique inférentielle

La *statistique inférentielle* est la science de la catégorisation ou de la prise de décisions concernant une population par l'utilisation d'informations provenant d'un échantillon extrait de celle-ci. C'est un processus de généralisation à une population non entièrement étudiée par l'utilisation de données mesurées sur un échantillon de cette population. La statistique inférentielle a l'avantage de quantifier le degré de certitude d'une inférence particulière.

Le but de cette section n'est pas d'enseigner la statistique inférentielle mais de passer en revue les différents éléments intervenant dans l'analyse des données d'une expérience en génie logiciel afin de donner au lecteur les bases nécessaires à la compréhension des analyses effectuées et présentées dans le chapitre 8. Nous ne reviendrons pas sur le problème de la sélection des variables déjà traité dans la section 3.2.4, ni sur les techniques d'extraction d'échantillons d'une population présentées dans la section 3.2.4. D'une manière générale, cette section est inspirée de l'ouvrage écrit par Boslaugh et Watters intitulé « Statistics In a Nutshell » [BW08].

4.1.1 Distribution de probabilité

En statistique inférentielle, on s'appuie souvent sur des présuppositions concernant la distribution des données. Une *distribution de probabilité théorique* définit par une formule la fréquence (le nombre de fois qu'une valeur apparaît) des valeurs possibles pour les points (éléments de l'ensemble des valeurs) de la distribution. Une distribution peut être continue ou discrète.

Ces distributions de probabilité théoriques sont utiles en statistique inférentielle car leurs propriétés et caractéristiques sont bien connues. Si la distribution de l'ensemble de données étudiées est raisonnablement proche de la distribution théorique, alors beaucoup de calculs sont simplifiés car ils peuvent être réalisés en faisant l'hypothèse que l'ensemble des données suit cette distribution théorique. De plus, grâce au théorème central limite (voir section 4.1.2), il est possible de supposer

que la distribution de la moyenne d'un échantillon suffisamment grand suit la distribution théorique normale (voir plus bas), et ce, même si la population source de l'échantillon n'est pas normalement distribuée.

La distribution normale

La *distribution normale* (ou loi normale) est une distribution continue. C'est la distribution théorique la plus utilisée, d'une part en raison du théorème central limite mais aussi parce qu'elle est une description raisonnable de la fréquence de beaucoup de variables continues. La fonction permettant d'affirmer qu'une variable aléatoire suit une distribution normale est :

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} \tag{4.1}$$

σ : écart-type
μ : moyenne

Il existe un nombre infini de distributions normales. Elles ont la même courbe de base mais diffèrent d'après leur moyenne et écart-type. Un exemple de 3 distributions normales différentes est présenté par la figure 4.1.

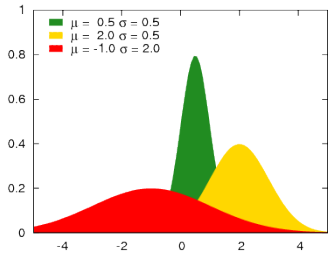


FIG. 4.1: Trois distributions normales.

La distribution normale de moyenne 0 et d'écart-type 1 est nommée *distribution normale standard*. Chaque distribution normale peut être convertie en distribution normale standard. Cela facilite les comparaisons entre distributions de moyennes et écarts-types différents.

Quels que soient les paramètres des distributions normales, elles partagent toutes les propriétés suivantes :

- Une parfaite symétrie.
- Une dispersion continue de $-\infty$ à $+\infty$.
- Une surface sous la courbe égale à 1.
- Une valeur unique pour le mode (la valeur la plus commune), la moyenne et la médiane.

Comme toutes les distributions normales ont la même courbe basique, il est possible de faire quelques suppositions concernant la distribution des données pour une distribution normale quelconque. Les règles empiriques sont :

- Environ 68 % des valeurs sont comprises dans la plage ± 1 écart-type
- Environ 95 % des valeurs sont comprises dans la plage ± 2 écarts-types
- Environ 99 % des valeurs sont comprises dans la plage ± 3 écarts-types

La connaissance de ces propriétés donne une moyen de juger si une valeur est atypique ou non par rapport aux autres valeurs de la population, surtout lorsqu'on considère la facilité de comparaison après conversion vers la distribution normale standard.

4.1.2 Le théorème central limite

Le *théorème central limite* établit que la distribution de la moyenne d'un échantillon est une approximation de la distribution normale, quelle que soit la population à l'origine de l'échantillon, pour peu que l'échantillon soit assez grand.

Le théorème peut être formulé de la manière suivante :

> Soient $X_1, \ldots, X_n$ un échantillon aléatoire d'une population suivant une loi de moyenne μ et de variance σ^2. Pour n suffisamment grand :
>
> $$\bar{X} \sim N\left(\mu, \frac{\sigma^2}{n}\right) \tag{4.2}$$
>
> même si la distribution des observations individuelles dans la population n'est pas normale.
>
> Le symbole $\sim$ représente la notion d'« approximativement distribué ». La formule peut donc être lue de la manière suivante : « la moyenne de X est approximativement normalement distribuée avec une moyenne de μ et une variance de $\frac{\sigma^2}{n}$ ».

Remarquons que cet énoncé [1] du théorème reste flou concernant la notion de « suffisamment grand ». La notion de suffisamment grand signifie qu'il existe un n à partir duquel la relation est vraie. L'intérêt pratique de ce théorème est que lorsqu'un échantillon a une distribution approximativement normale, les propriétés connues de cette distribution peuvent être utilisées, ce qui simplifie les calculs.

4.1.3 Test d'hypothèse

Les *tests d'hypothèse* sont fondamentaux en statistique inférentielle car ce sont eux qui nous permettent d'utiliser les méthodes statistiques afin de prendre une décision concernant un problème de la vie réelle sur base d'un échantillon de la population étudiée. Un test d'hypothèse implique 4 étapes :

1. Développer une hypothèse de recherche qui peut être testée mathématiquement.
2. Établir formellement les hypothèses nulle et alternative.
3. Décider du test à effectuer et faire les calculs.
4. Prendre une décision sur base du résultat.

L'hypothèse nulle : elle est notée H_0 et consiste à dire qu'il n'existe pas de différence entre les paramètres comparés ou que la différence observée est due au hasard (*e.g.*, $\mu_1 = \mu_2$). Elle est formulée dans le but d'être rejetée.

L'hypothèse alternative : elle est notée H_1 ou H_α voir encore H_{α_x} (x entier dans le cas où il y en a plusieurs) et est l'inverse de H_0 (*e.g.*, $\mu_1 \neq \mu_2$, $\mu_1 < \mu_2$ ou $\mu_1 > \mu_2$).

Rejeter H_0 revient alors à accepter H_1. L'hypothèse nulle et l'hypothèse alternative doivent être exclusives (aucun résultat ne peut satisfaire les 2 hypothèses) et exhaustives (chaque résultat possible doit satisfaire une seule des 2 conditions).

Le test d'hypothèse (*e.g.*, T-Test, Anova, etc.) exécuté sur les données collectées permet ensuite de décider du rejet ou non de H_0. Il faut noter que les conclusions des tests varient. Rejeter H_0 revient à accepter H_1 mais ne pas rejeter H_0 ne revient pas pour autant à accepter H_0. Cela signifie seulement qu'il n'y a pas assez d'évidence statistique permettant le rejet de H_0.

Le test d'hypothèse peut être unilatéral ou bilatéral. Un *test unilatéral* consiste à ne considérer qu'une des 2 possibilités pour l'hypothèse alternative (*e.g.*, soit $\mu_1 < \mu_2$, soit $\mu_1 > \mu_2$). Mais il

[1] La vraie forme du théorème est bien plus précise mais fait intervenir des notions bien plus avancées.

faut faire attention au fait que les hypothèses doivent toujours être exhaustives : si on prend $H_1 : \mu_1 < \mu_2$, il faut être absolument certain que l'inverse ne peut pas se produire. Un *test bilatéral* considérera les 2 possibilités : $H_1 : \mu_1 \neq \mu_2$.

Dans le cas où le test d'hypothèse nous permet de rejeter H_0, on dit que le résultat est *significatif*. Un processus statistique implique une décision concernant un niveau de probabilité au-delà duquel on considère que les données de l'échantillon sont assez robustes pour rejeter H_0. La *p-valeur* exprime la probabilité qu'ont les résultats d'être considérés comme significatifs alors qu'ils ne le sont pas. Des seuils arbitraires souvent utilisés en pratique sont $p < 0.05$, $p < 0.01$ ou $p < 0.001$ mais jamais plus que $p < 0.1$. Une p-valeur $p < 0.05$ signifie donc que la probabilité de se tromper en rejetant H_0 est de 5%. Le choix de la p-valeur dépend du risque pratique lié à la décision. Dans le monde académique, un seuil de 0.05 est généralement utilisé. Ce seuil est noté α.

Aujourd'hui les ordinateurs permettent un calcul aisé de la p-valeur. Lorsque l'on ne dispose pas d'ordinateur, il est difficile d'obtenir une p-valeur exacte. Il est alors nécessaire d'avoir recours à une table de référence.

Une fois le test effectué, il faut se positionner sur le rejet ou non de H_0 :

1. $p \geq \alpha$: H_0 n'est pas rejetée car le risque de rejeter H_0 alors qu'elle est vraie est trop élevé.
2. $p < \alpha$: H_0 est rejetée car le risque de rejeter H_0 alors qu'elle est vraie est inférieur au seuil choisi.

Comme nous l'avons dit, il y a tout de même une probabilité α de rejeter H_0 par erreur. Cette erreur est ce que l'on appelle l'*erreur de type I* ou *erreur* α. Il existe aussi l'*erreur de type II* ou *erreur* β qui correspond à ne pas rejeter H_0 alors que H_0 est effectivement fausse. Plus on décide qu'α est petit, moins l'erreur de type I se produira mais au détriment de l'erreur de type II.

4.1.4 Puissance d'un test statistique

La réciproque de l'erreur de type II, définie comme $1 - \beta$, est appelée *puissance* du test. Il s'agit de l'aptitude du test à rejeter H_0 correctement. Cette puissance est importante car son calcul permet de déterminer la probabilité qu'a le test d'être significatif et donc d'éviter de gaspiller des ressources. Le tableau 4.1 résume les situations possibles lors de la prise de décision suite à un test d'hypothèse.

Réalité / Décision	H_0 vraie	H_0 fausse
H_0 acceptée	Décision correcte	Erreur de type II (β)
H_0 rejetée	Erreur de type I (α)	Puissance du test ($1 - \beta$)

TAB. 4.1: Erreurs de type I, de type II et puissance d'un test statistique.

Il y a 4 éléments qui influencent la puissance d'un test :

1. Le seuil de signification α (une grande valeur augmente la puissance).
2. La différence réelle du paramètre considéré entre les populations comparées (plus elle est grande, plus la puissance l'est aussi).
3. La variabilité interne à la population du paramètre considéré (plus elle est grande, moins le test sera puissant).
4. La taille de l'échantillon (plus elle est grande, plus la puissance sera grande).

Le seul facteur que l'expérimentateur est en mesure de contrôler est la taille de l'échantillon. Mais le recrutement de sujets coûte. La puissance d'un test lui permet donc de prévoir une taille d'échantillon raisonnable : une puissance statistique et des coûts acceptables.

Bien que le concept de puissance d'un test statistique soit simple, en pratique, son calcul ne l'est pas. Dans la plupart des cas, la seule information dont le chercheur dispose est le seuil α.

Laurencelle [Lau07], dans un article qu'il qualifie lui-même d'« en partie polémique », propose quelques considérations utiles au chercheur pour ce calcul de la puissance statistique.

4.1.5 Transformation de données

Les *tests d'hypothèse paramétriques* sont les plus utilisés en pratique (voir section 4.1.3). Malheureusement, ils requièrent que l'échantillon étudié possède certaines caractéristiques (*e.g.*, indépendance des groupes, homogénéité des variances, etc.). Si l'échantillon ne possède pas ces caractéristiques, le chercheur a plusieurs possibilités pour tout de même analyser ses données. Soit il utilise des *tests d'hypothèese non paramétriques* qui requièrent moins de conditions mais qui sont plus faibles, soit il tente de transformer ses données de manière à ce qu'elles respectent les conditions des tests paramétriques qu'il veut exécuter. Nous allons discuter ici du cas où la distribution des données doit respecter une distribution normale car c'est le cas le plus courant et c'est le cas qui s'est présenté lors de l'analyse des données de notre expérience.

La première étape est évidemment de déterminer si les données respectent la distribution normale assez raisonnablement. Dans la positive, il ne faut rien faire. Dans la négative, il faut tenter une transformation (valide !) des données. Afin de déterminer si les données respectent la distribution normale, 2 approches sont possibles. La première est graphique et consiste à présenter les données, par exemple sous forme d'un histogramme, afin d'analyser la courbe générale. Cela permet également de détecter les valeurs solitaires et permet au chercheur expérimenté d'avoir une idée de la transformation à effectuer si nécessaire. La seconde méthode, *le test de normalité*, consiste à tester si les données repectent la distribution théorique normale (*e.g.*, le test de Kolmogorov-Smirnov). Il s'agit d'un test d'hypothèse dont le résultat permet de se positionner. Si le résultat est inférieur à un seuil α choisi, les résultats sont trop éloignés de la distribution normale et une transformation est nécessaire.

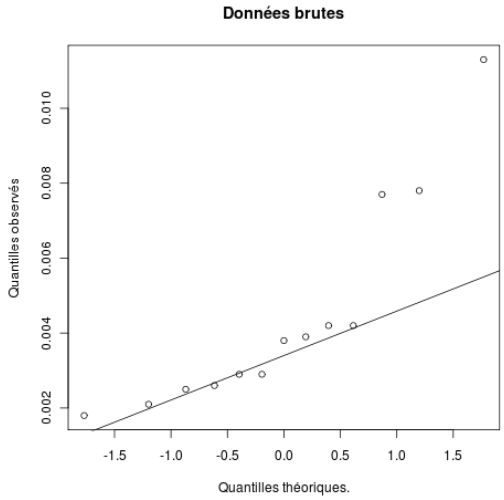
Deux transformations courantes sont effectuées en prenant la racine carrée de chacune des valeurs ou en prenant le logarithme. Il faut évidemment prendre des dispositions particulières concernant les valeurs négatives (*e.g.*, ajouter une constante à chaque valeur pour qu'elles soient toutes positives). Il faut ensuite refaire le test de normalité sur les nouvelles valeurs. En cas d'échec du test, il faut chercher une autre transformation ou utiliser les tests non paramétriques.

L'efficacité de la transformation n'est pas garantie et peut même créer plus de problèmes qu'elle n'en résout. Une transformation de données change l'unité des données et ce changement doit *impérativement* être pris en compte lors de l'interprétation. Certains calculs sont impossibles sur les données transformées ou n'ont pas de sens.

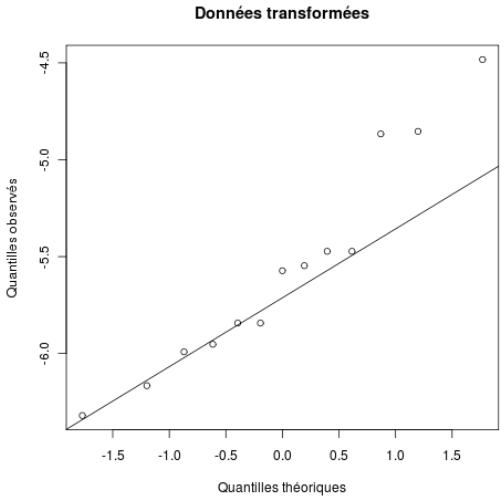
Afin de visualiser graphiquement si un échantillon respecte une distribution théorique, on utilise un *graphique quantile-quantile*. On estime une fonction de répartition F (fréquence) à partir des observations. Cette fonction est représentée sur l'axe des y. L'axe des x représente la fonction de répartition G de la distribution théorique. Pour chaque valeur y de l'observation, on lui fait correspondre une valeur x telle que : $x = G^{-1}F(y)$. On trace également la droite passant par le 1^{er} et 3^{e} quartile. Si F provient de G, les points devraient suivre cette droite.

La figure 4.2 illustre la façon dont une transformation de données peut adapter un échantillon pour qu'il respecte une distribution normale [2]. La figure 4.2a représente un échantillon ne respectant pas la loi normale selon le test de Kolmogorov-Smirnov tandis que la figure 4.2b représente un échantillon respectant la loi normale selon le test. Les données de la figure 4.2b sont les mêmes que celles de la figure 4.2a après une transformation logarithmique.

[2] Cette exemple à réellement été utilisé.



(a) Graphique quantile-quantile des données brutes avec la distribution normale.



(b) Graphique quantile-quantile des données après une transformation logarithmique avec la distribution normale.

FIG. 4.2: Illustration de l'utilité des transformations de données. L'axe des x représente les écarts à la moyenne en nombre d'écart-type σ.

4.2 Analyse de la variance : Anova

L'*analyse de la variance* (plus précisément le 2-way Anova) est le test d'hypothèse utilisé dans notre expérience. C'est une technique statistique qui compare plusieurs populations. Il en existe plusieurs versions permettant de répondre à des questions différentes. Pour exécuter cette technique, on utilise généralement un logiciel fournissant comme réponse ce que l'on appelle une table d'Anova. Le but de cette section n'est pas de fournir au lecteur les éléments nécessaires à la création de cette table mais de fournir ceux nécessaires à son interprétation.

La forme la plus simple est le 1-way Anova et est présentée dans la section 4.2.1. Il permet de déterminer si l'effet d'une variable indépendante sur une variable dépendante est différent entre plusieurs groupes indépendants. L'Anova peut être étendu pour prendre en compte plusieurs variables indépendantes ainsi que leurs interactions. On parle alors d'Anova factorielle. Le cas du 2-way Anova est présenté dans la section 4.2.2 et celui du 3-way Anova dans la section 4.2.3. Il existe d'autres extensions de l'Anova non présentées ici. Pour plus d'information, le lecteur peut consulter un ouvrage plus spécialisé (*e.g.*, [BW08]).

4.2.1 One-way Anova

Il s'agit d'une technique communément utilisée pour tester s'il existe une différence statistiquement significative entre 2 ou plusieurs groupes indépendants. Comme son nom ne le laisse pas présager, ce test n'analyse pas la variance mais compare les moyennes. L'hypothèse de base est qu'un ou plusieurs échantillons ont des moyennes différentes. Les hypothèses, nulle et alternative(s), se présentent comme ceci :

- $H_0 : \mu_1 = \mu_2 = \ldots = \mu_n$.
- H_1 : au moins 2 moyennes (et donc 2 populations) sont différentes.

Avant d'effectuer le test, il faut s'assurer que les données respectent les conditions de l'Anova :

- Indépendance des n échantillons.

– Normalité des n distributions.
– Homoscédasticité : les variances des n distributions sont homogènes.

Le test prend en compte la moyenne totale, les moyennes individuelles de chaque groupe, la variation totale, la variation entre les groupes et les variations internes aux groupes. Le rapport, en prenant en compte les degrés de liberté [3] de chaque variation, entre la variation interne et la variation externe est appelé F-test. Si la variation entre les échantillons est grande comparée à la variation dans les échantillons, on peut s'attendre à ce qu'elles soient originaires de populations différentes.

Prenons par exemple 4 groupes pour lesquels la valeur de la variable dépendante est donnée par le tableau 4.2 et la table d'Anova relative fournie par le logiciel R [4] est donnée par le tableau 4.3.

Test	Groupe 1	Groupe 2	Groupe 3
1	5.67	5.75	4.74
2	5.67	5.47	4.45
3	5.55	5.43	4.65
4	5.57	5.45	4.94

TAB. 4.2: Données d'exemple pour le 1-way Anova.

```
Analysis of Variance Table

Response: values
          Df  Sum Sq Mean Sq F value    Pr(>F)
Factor     2 2.05787 1.02893  45.239 2.015e-05 ***
Residuals  9 0.20470 0.02274
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

TAB. 4.3: Exemple de table d'Anova fournie par R.

La ligne intitulée « Factor » donne la variance inter-échantillons et la ligne intitulée « Residuals » donne la variance interne aux échantillons. La p-valeur de $2.015e^{-5}$ nous dit qu'il y a une probabilité de presque 99.998% pour que la variance inter-échantillons soit significativement plus grande que la variance interne aux échantillons. Il y a donc une évidence assez forte pour rejeter H_0 au profit de H_1. Il existe donc 2 groupes avec une moyenne différente.

Malheureusement, une fois H_0 rejetée, le test 1-way Anova ne nous dit pas quels sont les groupes avec une différence de moyenne significative, ni s'il n'y en a qu'un qui diffère des autres ou si les 3 moyennes sont différentes. C'est pourquoi, pour déterminer quels groupes sont différents, il faut effectuer un test supplémentaire. Pour plus d'informations, le lecteur peut consulter [BW08]

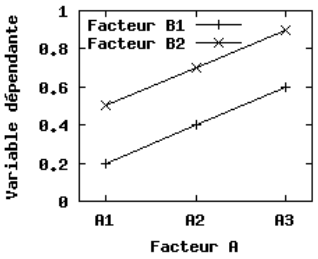
4.2.2 Two-way Anova

Le 2-way Anova est une extension du 1-way Anova et implique 2 variables indépendantes. Il permet non seulement de déterminer pour chacune des variables si elles ont un effet statistiquement significatif sur la variable dépendante (la moyenne) mais il permet également de déterminer si l'interaction des 2 variables indépendantes a un effet significatif. Il permet donc de déterminer si 2 facteurs agissant ensemble le font différemment que s'ils agissent séparément. Les hypothèses sont les mêmes que dans le cas du 1-way Anova.

Le test permet d'évaluer 2 facteurs et leurs interactions. Cela peut donner lieu à plusieurs situations telles que celles illustrées par la figure 4.3. Cette figure met également en relation un exemple de

[3] En statistique, le degré de liberté désigne le nombre de valeurs aléatoires qui ne peuvent être fixées par une équation. Dans l'équation $x + y + z = 12$, 2 variables peuvent être choisies librement. Le degré de liberté est donc 2. Si l'on choisit une valeur pour x et une pour y, la valeur de z est imposée par l'équation.

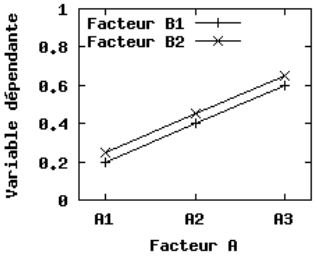
[4] `http://www.r-project.org/`



(a) Facteurs A et B significatifs

Source	DF	SS	MS	F	P
Facteur A	2	56.33	28.17	112.67	0.00
Facteur B	1	4.17	4.17	16.67	0.001
Interaction	2	0.33	0.17	0.67	0.53

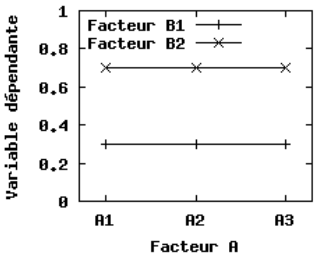
(b) Facteurs A et B significatifs



(c) Facteur A significatif

Source	DF	SS	MS	F	P
Facteur A	2	36.75	18.37	47.25	0.00
Facteur B	1	0.67	0.67	1.71	0.21
Interaction	2	0.08	0.04	0.11	0.9

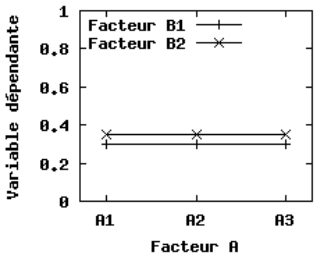
(d) Facteur A significatif



(e) Facteur B significatif

Source	DF	SS	MS	F	P
Facteur A	2	0.08	0.04	0.16	0.85
Facteur B	1	12.4	12.04	45.63	0.00
Interaction	2	0.08	0.04	0.16	0.85

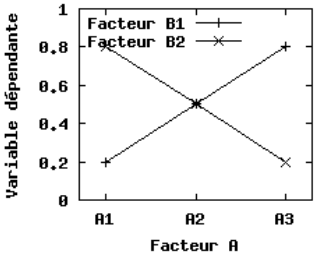
(f) Facteur B significatif



(g) Facteurs A et B non significatifs

Source	DF	SS	MS	F	P
Facteur A	2	0.08	0.04	0.08	0.93
Facteur B	1	0.37	0.37	0.69	0.42
Interaction	2	0.75	0.37	0.69	0.51

(h) Facteurs A et B non significatifs



(i) Interaction significative

Source	DF	SS	MS	F	P
Facteur A	2	0.08	0.04	0.16	0.85
Facteur B	1	0.37	0.37	1.42	0.25
Interaction	2	16.75	8.37	31.74	0.00

(j) Interaction significative

FIG. 4.3: Exemples de graphiques d'interactions avec des tables d'Anova typiques (adapté de [RP07]).

table de 2-way Anova en rapport avec les graphiques. Les graphiques sont construits en prenant les moyennes relatives des différents groupes :

1. Les facteurs A et B sont significatifs (figure 4.3a et tableau 4.3b).
2. Le facteur A est significatif mais pas le B (figure 4.3c et tableau 4.3d).
3. Le facteur B est significatif mais pas le A (figure 4.3e et tableau 4.3f).
4. Ni le facteur A ni le B ne sont significatifs (figure 4.3g et tableau 4.3h).
5. L'interaction entre A et B est significative (figure 4.3i et tableau 4.3j).

4.2.3 Tree-way Anova

Le 3-way Anova est une extension simple du 2-way Anova. Plutôt que de traiter 2 facteurs, ce test en prend en compte 3. La table d'Anova fournit dès lors 7 informations :

1. Le niveau de signification du facteur A.
2. Le niveau de signification du facteur B.
3. Le niveau de signification du facteur C.
4. Le niveau de signification de l'interaction entre les facteurs A et B.
5. Le niveau de signification de l'interaction entre les facteurs A et C.
6. Le niveau de signification de l'interaction entre les facteurs B et C.
7. Le niveau de signification de l'interaction entre les facteurs A, B et C.

L'interprétation de la table ne pose pas de difficultés supplémentaires par rapport au 2-way Anova. La représentation graphique se fait en 3 dimensions et les hypothèses de base sur les différents groupes sont les mêmes. Il est bien entendu possible d'étendre encore le test pour prendre en compte plus de variables indépendantes.

Comme nous l'avons vu, la technique de l'Anova est une technique puissante, beaucoup utilisée dans la recherche, dont les résultats sont faciles à interpréter, mais elle reste contraignante : indépendance des échantillons, normalité des données et homogénéité des variances.

4.3 Éléments techniques

Cette section présente certains éléments utiles à la compréhension de la méthode utilisée pour la création des groupes pour l'expérience. Cette méthode est expliquée plus loin dans la section 8.3. La section 4.3.1 introduit la distance de Hausdorff et la section 4.3.2 donne une très brève introduction à la classification et plus particulièrement à la technique Agnes.

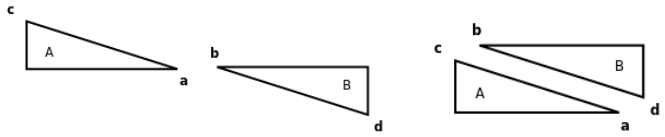
4.3.1 Distance de Hausdorff

Grégoire et Bouillot [GB98] expliquent clairement l'intérêt de cette distance. Lorsque l'on parle de distances, nous entendons généralement la plus courte distance. Si on dit qu'1 point x est à une distance d d'un polygone P, on suppose généralement que d est la distance séparant x du point le plus proche de P. La même logique s'applique aux polygones. Si 2 polygones, A et B, sont à une distance d l'un de l'autre, nous comprenons généralement que d est la distance entre le point de A le plus proche de B et le point de B le plus proche de A. C'est la fonction minimum :

$$D(A, B) = \min_{a \in A}\{\min_{b \in B}\{d(a, b)\}\} \tag{4.3}$$

Mais cette définition est insatisfaisante pour certaines applications. Si l'on analyse la figure 4.4a, on constate que les triangles sont proches l'un de l'autre si on prend en considération la distance minimale : sommets a et b. Par contre, si l'on considère les sommets c et d, ils ne sont pas proches.

La distance minimale de la figure 4.4b est (environ) la même que pour la figure 4.4a alors qu'intuitivement, ces 2 polygones sont plus proches. Le concept de distance la plus courte souffre donc, dans le cas des polygones, d'un manque d'information.



(a) La plus courte distance est $d(a, b)$ et ne prend pas en compte $d(c, d)$.

(b) La plus courte distance est $d(a, b)$ ou $d(c, d)$.

FIG. 4.4: Illustration de la faiblesse de la distance minimum entre 2 polygones (adapté de [GB98]) : les distances minimales sont égales pour les deux figures alors qu'intuitivement la distance entre les 2 polygones ne l'est pas.

La distance

La *distance de Hausdorff* [Rot91] entre 2 ensembles finis [5] de nombres $A = \{a_1, a_2, \ldots, a_n\}$ et $B = \{b_1, b_2, \ldots, b_m\}$ est définie par :

$$h(A, B) = \max\{\max_{a \in A} \min_{b \in B} d(a, b), \max_{b \in B} \min_{a \in A} d(b, a)\} \tag{4.4}$$

Il s'agit donc de la plus petite distance h de sorte que chaque valeur de A ait une valeur de B où $d(a, b) < h$ et que chaque valeur de B ait une valeur de A où $d(b, a) < h$. La distance de Hausdorff peut être définie pour des ensembles de points de 2 dimensions (ce qui nous intéresse) ou plus. Dans R^2, la distance $d(a, b)$ est alors remplacée par la distance euclidienne (ou toute autre distance appropriée) entre les points a et b.

Pour illustrer la valeur de cette distance, reprenons la figure 4.4. Les figures 4.5c et 4.5a illustrent leurs distances $h(A)$ et $h(B)$ possibles. La distance de Hausdorff h sera $\max\{h(A), h(B)\}$. Le lecteur attentif remarquera que la distance de Hausdorff s'applique également dans le cas où les 2 polygones se recouvrent : figure 4.5b.

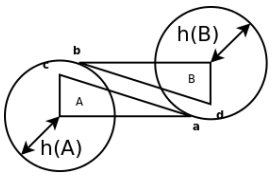
Notons que la distance de Hausdorff respecte les 4 exigences d'une fonction de distance :
D1 : $h(i, j) \geq 0$
D2 : $h(i, i) = 0$
D3 : $h(i, j) = h(j, i)$
D4 : $h(i, j) \leq h(i, k) + h(k, j)$
De plus, cette distance peut être utilisée comme mesure car [DJ94] :

$$(h(i, j) = 0) \Rightarrow (i = j) \tag{4.5}$$

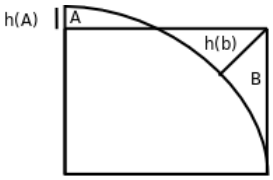
Utilisation

Cette technique est utilisée dans des domaines variés : la localisation de robots [KNW99] ; la reconnaissance faciale [JKF01] ; la recherche textuelle dans des images de documents (documents scannés par exemple) sur base de l'image des mots [AK08] ; la recherche d'images par le contenu (Content Based Image Retrieval (CBIR)) [PLL08] ; etc.

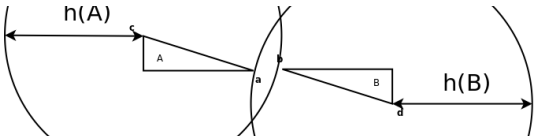
[5] Le maximum d'un ensemble n'existe pas toujours. Quel est le maximum de l'ensemble des réels tels que $x < 1$?



(a) La distance de Hausdorff relative à la figure 4.4b (adapté de [GB98]).



(b) La distance de Hausdorff avec 2 polygones se recouvrant (Adapté de [Wik09]).



(c) La distance de Hausdorff relative à la figure 4.4a (adapté de [GB98]).

FIG. 4.5: Illustration de la distance de Hausdorff : $h = max\{h(A), h(B)\}$ dans les 3 cas.

4.3.2 Classification

Cette section est inspirée de l'ouvrage de KAUFMAN et ROUSSEEUW « Finding Groups in Data : An Introduction to Cluster Analysis » [KR90] que le lecteur peut consulter pour un complément d'informations.

Classer les objets similaires en groupes est une activité humaine importante. Par le passé, les groupements étaient réalisés de manière subjective en se basant sur le jugement du chercheur. Mais la nécessité de classification dans les cas où plus de 3 dimensions entrent en jeux et l'objectivité demandée par les sciences actuelles ont donné lieu à des procédures de classifications automatiques.

Les données à grouper peuvent être arrangées selon 2 méthodes. La première est selon une matrice $n \times m$ où les lignes correspondent aux objets et les colonnes correspondent aux attributs. Une telle matrice est appelée 2-modes car les lignes et colonnes sont différentes. La seconde est une collection de mesures de proximité définie pour chaque paire d'objets. Ces mesures permettent la construction d'une matrice $n \times n$ appelée 1-mode puisque les entités des lignes et colonnes sont les mêmes objets. On peut considérer 2 types de mesure de proximité : une mesure de dissimilarité ou une mesure de similarité. Les variables peuvent être définies sur les différents types d'échelle définis à la section 3.1.4. Il existe également des techniques particulières pour le traitement des variables binaires.

Mesures de proximité

Les *mesures de proximité* sont généralement utilisées car elles sont moins contraignantes et plus générales que les mesures. Elles peuvent même être subjectives (*e.g.*, un sujet donne une estimation de la ressemblance entre 2 objets).

Une *mesure de dissimilarité* mesure l'éloignement entre 2 objets. L'arrangement de ces coefficients de dissimilarité en une matrice $n \times n$ est appelé *matrice de dissimilarité*. Ce coefficient est un nombre non-négatif $d(i, j)$ qui est petit (proche de 0) lorsque i et j sont proches et qui devient grand lorsque i et j sont fort différents. On suppose généralement que la dissimilarité est symétrique et que la dissimilarité d'un objet avec lui-même vaut 0. Bien que ce ne soit généralement pas nécessaire, on considère qu'une mesure du coefficient de dissimilarité satisfait aux propriétés (D1), (D2), et (D3) (vues à la page 36) mais pas à l'équation triangulaire (D4). Certains algorithmes de groupement s'appliquent directement à une matrice de dissimilarité.

Une *mesure de similarité* mesure la ressemblance entre 2 objets. Plus 2 objets i et j sont proches, plus le coefficient de similarité $s(i, j)$ est grand et inversement. Une telle mesure prend généralement une valeur comprise entre 0 et 1. Les conditions suivantes sont généralement valables pour chaque objet i et j :

S1 : $0 \leq s(i, j) \leq 1$
S2 : $s(i, j) = 1$
S3 : $s(i, j) = s(j, i)$

Encore une fois, les coefficients $s(i, j)$ peuvent être arrangés en une matrice $n \times n$, appelée *matrice de similarité*. Les matrices de similarité et de dissimilarité sont également appelées *matrice de proximité*.

Famille de méthode

La grande majorité des techniques de regroupement peut être classée selon 2 types : les méthodes de partitionnements et les méthodes hiérarchiques. Les *méthodes de partitionnements* construisent k groupes de données en respectant les 2 règles suivantes :

1. Chaque groupe contient au moins 1 objet.
2. Chaque objet appartient à au moins 1 groupe.

Ces conditions impliquent qu'il n'existe pas plus de groupes que d'objets $k \leq n$.

Ce type de méthode est utilisé pour classer les objets en k groupes où k est fixé et peuvent être adaptées pour que le k le plus adapté soit déterminé automatiquement.

Les *méthodes hiérarchiques* ne construisent pas une partition de k groupes mais s'occupent de toutes les valeurs de k en une fois. Si $k = 1$ tous les objets sont dans le même groupe, si $k = n$ chaque objet est dans son groupe. Les valeurs entre 2 ($k = 2, 3, \ldots, n - 1$) sont couvertes par transition. La différence entre $k = r$ et $k = r + 1$ est qu'un des groupes est séparé en 2. Les méthodes hiérarchiques comprennent les *méthodes agglomératives* qui assemblent 2 groupes pour en former 1 autre et passer de k groupes à $k - 1$ et les *méthodes divisives* qui divisent 1 groupe en 2 pour passer de k groupes à $k + 1$. Enfin, ces méthodes sont plus rigides que les méthodes hiérarchique car elles ne peuvent revenir à une étape précédente et une division/association faite ne peut être défaite.

Agglomerative Nesting (AGNES)

AGNES est une technique de classification hiérarchique agglomérative. L'algorithme construit une hiérarchie sous forme d'un arbre contenant implicitement les groupes pour toutes les valeurs possibles de k. L'algorithme prend, comme entrée, une matrice de dissimilarité. Afin d'illustrer la nécessité d'un algorithme, prenons comme exemple la matrice du tableau 4.4. Bien que cet exemple soit extrêmement simple, un algorithme permettant de dégager les structures sous-jacentes est utile.

	a	b	c	d	e
a	0.0	2.0	6.0	10.0	9.0
b	2.0	0.0	5.0	9.0	8.0
c	6.0	5.0	0.0	4.0	5.0
d	10.0	9.0	4.0	0.0	3.0
e	9.0	8.0	5.0	3.0	0.0

TAB. 4.4: Illustration de la nécessité d'algorithmes de classification.

L'étape 0 consiste à faire de chaque objet un groupe. L'étape suivante fusionne les 2 groupes les plus proches. Il suffit de trouver l'entrée la plus petite (hors diagonale) de la matrice et de

fusionner les 2 objets correspondants. Dans le cas où plusieurs entrées ont la même valeur, il suffit d'en choisir une aléatoirement. Une fois les 2 objets fusionnés, il faut mettre à jour la matrice et déterminer les coefficients de dissimilarité du nouveau groupe. Il existe plusieurs manières de calculer ces coefficients. Celle utilisée par AGNES est simple :

Soient les 2 groupes R et Q et leurs cardinalités $\mid R \mid$ et $\mid Q \mid$. Leur dissimilarité $d(R, Q)$ est définie comme la dissimilarité moyenne des dissimilarités $d(i, j)$, où $i \in R$ et $j \in Q$:

$$d(R, Q) = \frac{1}{\mid R \mid\mid Q \mid} \sum_{i \in R, j \in Q} d(i, j) \tag{4.6}$$

Reprenons l'exemple du tableau 4.4 pour illustrer l'algorithme.

Étape 0 : elle consiste à faire de chaque objet un groupe.

Étape 1 : le coefficient le plus faible de la matrice est celui correspondant au groupe {a} et {b} qui vaut 2.0. On va donc fusionner ces 2 groupes en 1 groupe {a,b}. Les coefficients sont calculés par :

- $d(\{a, b\}, \{c\}) = \frac{1}{2}[d(a, c) + d(b, c)] = 5.5$
- $d(\{a, b\}, \{d\}) = \frac{1}{2}[d(a, d) + d(b, d)] = 9.5$
- $d(\{a, b\}, \{e\}) = \frac{1}{2}[d(a, e) + d(b, e)] = 8.5$

La matrice devient donc :

	$\{a, b\}$	$\{c\}$	$\{d\}$	$\{e\}$
$\{a, b\}$	0.0	5.5	9.5	8.5
$\{c\}$	5.5	0.0	4.0	5.0
$\{d\}$	9.5	4.0	0.0	3.0
$\{e\}$	8.5	5.0	3.0	0.0

Étape 2 : on fusionne les groupes $\{d\}$ et $\{e\}$:

- $d(\{d, e\}, \{c\}) = \frac{1}{2}[d(d, c) + d(e, c)] = 4.5$
- $d(\{d, e\}, \{a, b\}) = \frac{1}{4}[d(d, a) + d(d, b) + d(e, a) + d(e, b)] = 9$

	$\{a, b\}$	$\{c\}$	$\{d, e\}$
$\{a, b\}$	0.0	5.5	9.0
$\{c\}$	5.5	0.0	4.5
$\{d, e\}$	9.0	4.5	0.0

Étape 3 : on fusionne les groupes $\{c\}$ et $\{d, e\}$:

- $d(\{c, d, e\}, \{a, b\}) = \frac{1}{6}[d(c, a) + d(c, b) + d, d, a) + d(d, b) + d(e, a) + d(e, b)] = \frac{47}{6} \cong 7.83$

	$\{a, b\}$	$\{c, d, e\}$
$\{a, b\}$	0.00	7.83
$\{c, d, e\}$	7.83	0.00

Étape 4 : enfin la dernière étape consiste à fusionner les 2 derniers groupes pour obtenir un groupe formé de tous les objets.

Il est possible de représenter graphiquement le résultat sous forme d'un arbre. De plus, la valeur minimale de dissimilarité au fur et à mesure que le nombre de groupes diminue est monotone. Dans notre exemple, elles valent : 2.0, 3.0, 4.5 et finalement 7.83.

La formule 4.6 peut être adaptée pour que l'on n'ait plus besoin de la matrice de départ à chaque étape :

Soient 2 groupes A et B que l'on fusionne pour former un groupe R :

$$\begin{aligned}
d(R, Q) &= \frac{1}{\mid R \mid\mid Q \mid} \sum_{i \in R, j \in Q} d(i, j) \\
&= \frac{1}{\mid R \mid\mid Q \mid} \sum_{i \in A, j \in Q} d(i, j) + \frac{1}{\mid R \mid\mid Q \mid} \sum_{i \in B, j \in Q} d(i, j) \\
&= \frac{\mid A \mid}{\mid R \mid} \left(\frac{1}{\mid A \mid\mid Q \mid} \sum_{i \in A, j \in Q} d(i, j) \right) + \frac{\mid B \mid}{\mid R \mid} \left(\frac{1}{\mid B \mid\mid Q \mid} \sum_{i \in B, j \in Q} d(i, j) \right)
\end{aligned}$$

Nous obtenons une formule qui permet de calculer la matrice de disimilarité à partir de la matrice de l'étape précédente :

$$d(R,Q) = \frac{\mid A \mid}{\mid R \mid} d(A,Q) + \frac{\mid B \mid}{\mid R \mid} d(B,Q) \tag{4.7}$$

Chapitre 5

Patrons de conception

Un patron de conception exprime une solution éprouvée à un problème de conception pour les logiciels orientés-objets. Ils furent introduits par Erich Gamma, Richard Helm, Ralph Johnson et John Vlissides dans leur ouvrage « Design Patterns : Elements of Reusable Object-Oriented Software » [1]. Cet ouvrage présente 23 patrons de conception dits classiques. Ils sont indépendants d'un domaine particulier mais les auteurs précisent que chaque domaine a ses patrons et qu'il serait intéressant de les cataloguer. Bruce Eckel parle de ces patrons de conception comme : *« Probably the most important step forward in object-oriented design is the « design patterns » movement[...] » [« Le mouvement des patrons de conception est probablement le pas en avant le plus important en conception orientée-objet[...] »].*

La section 5.1 explique le concept de patron de conception. Les spécialistes n'étant pas unanimes sur la notion de patron de conception, nous reprenons l'avis des pionniers du domaine. La section 5.2 donne un aperçu du catalogue des 23 patrons de conception classiques. Dans la section 5.3 nous revenons plus en détail sur les qualités attendues des patrons de conception. La section 5.4 introduit les 2 patrons utilisés dans notre expérience. Enfin, la section 5.5 donne un aperçu de quelques études empiriques menées sur le lien existant entre les patrons de conception et la maintenance de logiciels. Beaucoup d'éléments de ce chapitre sont directements tirés de l'ouvrage qui a introduit les patrons de conception [GHJV95].

5.1 Définition d'un patron de conception

Les patrons de conception ont été inspirés par les patrons de conception architecturaux de Christopher Alexander [AIS78]. Selon lui, chaque patron de conception décrit un problème qui se présente encore et encore dans notre environnement, il décrit ensuite le cœur de la solution, de sorte qu'il est possible d'utiliser la solution 1 million de fois, sans faire la même chose 2 fois. Il parlait de bâtiments et villes, mais cette vision reste vraie pour les patrons de conception qui nous intéressent.

Un patron de conception est décrit par 4 éléments :

Le nom : un mot ou deux peuvent être utilisés pour décrire à la fois un problème, sa solution et ses conséquences. Nommer un patron de conception nous permet d'en parler avec les collègues, dans la documentation et facilite la communication. Le nom permet également de réfléchir à un niveau d'abstraction plus élevé.

Le problème : il décrit *quand* utiliser le patron, il explique le problème résolu et le contexte. Il peut décrire des problèmes tels que la manière de représenter un algorithme en objets ou

[1] Ces quatre auteurs ont reçu le surnom de Gang of Four (GoF).

les structures symptomatiques d'une conception rigide. Il peut parfois inclure une liste de conditions nécessaires pour donner un sens à l'utilisation du patron.

La solution : elle décrit les éléments de la conception, leurs relations, responsabilités et collaborations. Elle ne donne pas une conception concrète ou une implémentation, car un patron agit comme un modèle pouvant être utilisé dans différentes situations. À la place, le patron fournit une description abstraite d'un problème de conception et l'agencement général des éléments pour le régler.

Les conséquences : c'est le résultat de l'application du patron. Il est important pour l'évaluation des alternatives de conception et pour comprendre le pour ou contre de l'utilisation du patron. Les conséquences concernent généralement l'espace et le temps, voire des problèmes dus au langage utilisé. Les conséquences aident à la compréhension des impacts sur la flexibilité, l'extensibilité et la portabilité du système.

Il faut noter que le point de vue affecte l'interprétation personnelle de ce qu'est un patron de conception puisque le patron de conception d'une personne peut n'être qu'une brique primitive du point de vue d'une autre personne.

5.2 Le catalogue du GOF

Comme il y a beaucoup de patrons de conception, il est intéressant de bien les organiser. Les 23 patrons classiques sont organisés selon 2 critères. Le premier critère est le but du patron :

- Création : concerne le processus de création des objets.
- Structure : concerne la composition des classes et objets.
- Comportement : caractérise le moyen par lequel les classes et objets interagissent et distribuent les responsabilités.

Le second critère concerne la portée du patron :

- Classe : concerne les relations entres les classes et leurs sous-classes. Ces relations sont établies via l'héritage et sont fixées lors de la compilation.
- Objet : concerne les relations entre objets. Elles sont plus dynamiques et peuvent changer à l'exécution.

Presque tous les patrons utilisent l'héritage. Les seuls patrons « de classe » sont ceux qui se focalisent sur les relations de classes. Ils sont clairement minoritaires.

But / Portée	Création	Structure	Comportement
Classe	Fabrique (Factory Method)	Adaptateur	Interpréteur Modèle de méthode
Objet	Fabrique Abstraite Monteur (Builder) Prototype Singleton	Adaptateur Pont Objet Composite Décorateur Facade Proxy	Chaîne de responsabilités Commande Itérateur Médiateur Memento Poids-mouche Observateur État Stratégie Visiteur

TAB. 5.1: Le catalogue des 23 patrons de conception classiques.

Le tableau 5.1 présente le catalogue des 23 patrons de conception classiques. Les patrons de création de classe sous-traitent une partie de la création des objets aux sous-classes tandis que les patrons de création d'objets la sous-traitent à d'autres objets. Les patrons structurels de classes utilisent l'héritage pour la composition de classes tandis que les patrons structurels d'objets décrivent une manière d'assembler les objets. Les patrons comportementaux de classes utilisent l'héritage pour

décrire un algorithme et le flux d'exécution, tandis que les patrons comportementaux d'objets décrivent comment un groupe d'objets coopère pour réaliser une tâche qu'un simple objet ne peut réaliser. Ce n'est pas la seule classification possible.

5.3 Qualités attendues d'un patron de conception

Une conception basée sur les patrons de conception possède, selon la littérature, de grandes qualités : flexibilité, robustesse, ré-utilisabilité, facilitation de communication, adaptabilité, etc. Rares sont les qualités qui ne leur ont pas été attribuées. D'une manière générale, il est admis que les patrons de conception améliorent la qualité des logiciels orientés-objets [GHJV95], [Ven05]. Néanmoins, certaines études [MB01], [Wen01] ou [Cli96] suggèrent tout de même que la qualité des logiciels n'est pas toujours améliorée.

Cline [Cli96] précise que pour tirer profit des patrons de conception, le cycle de vie entier du logiciel doit être adapté, autrement les bénéfices des patrons sont gaspillés. Il explique également que les classifications existantes ne sont utiles qu'aux connaisseurs des patrons et non à leur apprentissage et que certains patrons sont inutilement difficiles à apprendre.

Wendorff [Wen01], après avoir étudié l'utilisation des patrons de conception dans un gros système commercial, conclut que leur application peut parfois mener à des résultats négatifs. Selon lui, cela vient d'applications inappropriées. Soit parce que le programmeur ne les a pas correctement compris, soit parce que le patron utilisé n'est pas en accord avec les exigences du projet. Il ne faut donc pas hésiter à retirer des patrons du code, car leur coût peut être supérieur à leur bénéfice et leur utilisation ne doit pas être forcée.

McNatt et Bieman [MB01] critiquent l'utilisation des compositions de patrons de conception. La maintenabilité et la ré-utilisabilité sont fournies par des logiciels modulaires à faible couplage et avec une abstraction des détails afin d'éviter que la modification d'un composant n'affecte un autre. Selon leur étude, les compositions de patrons de conception le plus souvent utilisées dans la littérature et l'industrie corroborent cette vision et leur utilisation n'est pas favorable à ce couplage faible.

Sur base de la littérature, Khomh et Guéhéneuc ont établi un questionnaire [KG08] reprenant un ensemble d'attributs de qualité pertinent pour les patrons de conception et la qualité d'un système. Leur but était d'évaluer la qualité des patrons de conception individuellement par un sondage. Selon eux, les 10 qualités attendues d'un patron de conception sont :

- Relativement à la conception :
  - Extensibilité : la mesure dans laquelle la conception du système peut être étendue.
  - Simplicité : la mesure dans laquelle la conception du système peut être comprise avec facilité.
  - Ré-utilisabilité : la mesure dans laquelle une partie de la conception (ou une sous-partie d'une conception) peut être ré-utilisée dans une autre conception.
- Relativement à l'implémentation :
  - Apprentissage : la mesure dans laquelle le code source d'un système est simple à apprendre.
  - Compréhensibilité : la mesure dans laquelle le code source d'un système peut être compris sans difficultés.
  - Modularité : la mesure dans laquelle les implémentations des fonctions d'un système sont indépendantes les unes des autres.
- Relativement à l'exécution :
  - Généralité : la mesure dans laquelle le système peut opérer une grande variété de fonctions à l'exécution.
  - Modularité à l'exécution : la mesure dans laquelle les fonctions d'un système sont indépendantes les unes des autres à l'exécution.

- Évolutivité : la mesure dans laquelle le système peut faire face à une montée en charge à l'exécution.
- Robustesse : la mesure dans laquelle un système continue de fonctionner correctement dans des conditions anormales.

La conclusion du sondage est que certains patrons de conception amoindrissent certains attributs de qualité logicielle et n'améliorent même pas la ré-utilisabilité du tout. Il faut noter que Gamma *et al.* [GHJV95] dans leur ouvrage peuvent parler de ré-utilisabilité du point de vue des idées et que cette étude la considère aussi du point de vue du code. Certains principes de base peuvent également ne pas être aussi bons qu'il n'y paraît, n'étant pas nécessairement gage de systèmes de qualité.

5.4 Patrons de conception utilisés dans l'expérience

Commençons par préciser la notion d'interface et de couplage. La notion d'*interface* telle qu'utilisée par les patrons de conception n'est pas celle du langage Java. Il s'agit de toute classe ne pouvant pas être instanciée directement mais pouvant fournir un comportement par défaut. Une classe abstraite Java est dès lors considérée comme une interface. Un *couplage* fort signifie qu'un changement dans une composante engendre beaucoup de changements dans d'autres composantes.

Les 2 patrons de conception utilisés dans notre expérience sont Observateur et Objet composite. Leur sélection est discutée dans la section 7.2.6.

5.4.1 Observateur

Le patron de conception Observateur est un patron bien connu des programmeurs au moins de manière inconsciente. En effet, il est beaucoup utilisé dans les interfaces graphiques. La fenêtre graphique devant « observer » les changements d'états de l'objet qu'elle affiche.

But : définir une relation « un à plusieurs » entre des objets, de sorte que lorsqu'un objet change d'état, tous ses objets dépendants reçoivent un avertissement et soient mis à jour automatiquement.

Motivation : un effet de bord commun de la partition d'un système en une collection de classes qui coopèrent est la nécessité de conserver une consistance entre les objets liés. Il ne faut pas les coupler fortement, car cela réduit leur ré-utilisabilité. Le patron de conception Observateur décrit comment établir ce type de relation. Les objets clés sont le *sujet* et les *observateurs*. Un sujet peut avoir un nombre quelconque d'observateurs dépendants. Tous les observateurs reçoivent une notification lorsque le sujet change d'état. En réponse, chaque observateur va faire une requête de synchronisation de son état avec celui du sujet.

Applications : on peut utiliser ce patron de conception dans 3 situations :

1. Lorsqu'une abstraction comporte 2 aspects, l'un dépendant de l'autre. Encapsuler ces aspects dans des objets différents permet de les faire varier et de les réutiliser indépendamment.
2. Lorsqu'une modification d'un objet nécessite une modification d'autres objets et que le nombre de ces objets à modifier ou leur provenance sont inconnus.
3. Lorsqu'un objet doit être capable d'envoyer un avertissement à d'autres objets sans pouvoir faire de supposition sur ce que sont ces objets. En d'autres mots, lorqu'on ne veut pas coupler fortement ces objets.

Structure : la structure du patron est illustrée par un diagramme de classes à la figure 5.1.

Participants :

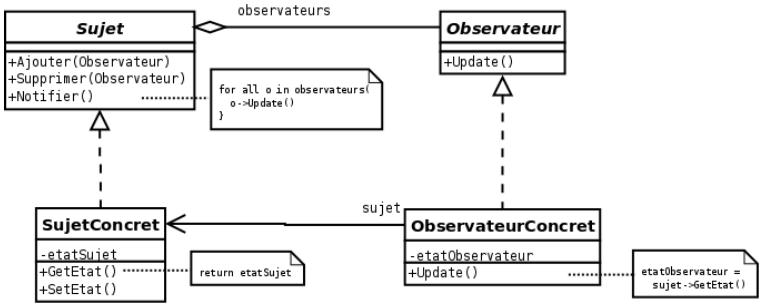


FIG. 5.1: Diagramme de classes du patron de conception Observateur (adapté de [GHJV95])
Sujet et Observateur sont 2 interfaces.

Sujet : connaît ses observateurs ; un nombre quelconque d'observateurs peut observer un sujet ; fournit une interface pour ajouter ou supprimer des objets observateurs.

Observateur : définit et met à jour l'interface pour les objets devant recevoir une notification des changements chez le sujet.

SujetConcret : contient l'état qui intéresse les objets observateurConcret ; envoie une notification à ses observateurs lorsque son état change.

ObservateurConcret : conserve une référence à l'objet sujetConcret ; contient l'état qui doit rester consistant avec celui du sujet ; implémente l'interface de mise à jour de l'observateur afin de conserver son état consistant avec celui du sujet.

Collaboration : la collaboration est illustrée par un diagramme de collaboration à la figure 5.2. Il est intéressant de noter que c'est un observateur qui est l'origine du changement d'état du sujet. Cependant celui-ci attend l'avertissement du sujet pour faire sa mise à jour. L'envoi des avertissements n'est pas nécessairement initié par le sujet. Un observateur ou un autre objet peut en être à l'origine.

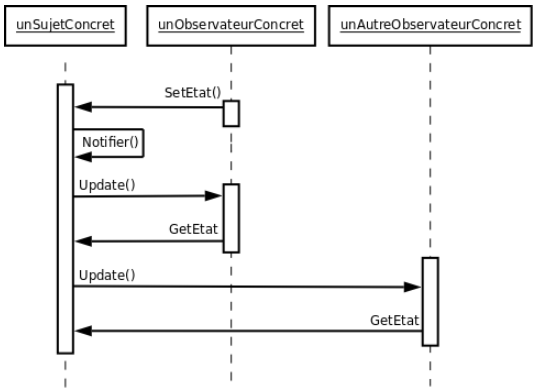


FIG. 5.2: Diagramme de collaboration de l'Observateur (adapté de [GHJV95]).

Conséquences : le patron de conception Observateur permet de faire varier les sujets et observateurs indépendamment. Il est possible de ré-utiliser les sujets sans ré-utiliser les observateurs. Il est possible d'ajouter des observateurs sans modifier le sujet ou d'autres observateurs.
Le patron implique également les avantages et responsabilités suivantes :

1. *Un couplage abstrait entre le sujet et les observateurs.* La seule chose dont le sujet est au courant est qu'il a une liste d'objets conformes à l'interface Observateur. Il ne connaît aucune des classes concrètes des observateurs, ce qui limite le couplage au minimum. Cela permet même aux sujets et observateurs d'appartenir à des couches d'abstraction différentes du système.

2. *Un support pour l'émission de communication.* L'avertissement du sujet envoie automatiquement un message à tous les objets intéressés inscrits. Le sujet ne se soucie pas du nombre d'objets intéressés, sa seule responsabilité est d'envoyer un avertissement à ses observateurs. Cela permet d'ajouter/supprimer un observateur à n'importe quel moment et laisse la possibilité à l'observateur d'ignorer le message.
3. *Gestion des mises à jour inattendues.* Comme les observateurs n'ont pas conscience de la présence de leurs collègues, ils sont extrêmement aveugles. Une opération apparemment anodine sur le sujet peut entraîner une cascade de mises à jour des observateurs et de leurs objets dépendants. Le simple protocole de mise à jour ne fournit aucun détail sur ce qui a changé chez le sujet. Sans mécanisme additionnel pour aider les observateurs à découvrir ce qui a changé, cela peut devenir compliqué pour eux.

5.4.2 Objet composite

But : Le but de l'Objet composite est de composer les objets en une structure d'arbre pour représenter des touts et leurs parties sous forme hiérarchique.

Motivation : les applications de dessin permettent aux utilisateurs de créer des diagrammes complexes à partir de composants simples. L'utilisateur peut assembler les composants pour créer de nouveaux composants plus complexes, pouvant à leur tour être assemblés pour créer des composants encore plus complexes.

Les composants primitifs et les composés sont traités de la même manière par l'utilisateur. Il ne faut donc pas distinguer ces 2 types d'objets dans le code sous peine de le complexifier. L'objet composite décrit comment mettre en œuvre la composition récursive de sorte que le client n'ait pas à faire la distinction entre les objets.

La clé de ce patron tient dans une classe abstraite qui représente les 2 types de composants : primitif et composé. Cette classe déclare toutes les opérations que chaque objet composé partage, telles que les opérations d'accès et de gestion des sous-éléments (les enfants). La figure 5.3 illustre un objet composite typique avec sa structure récursive.

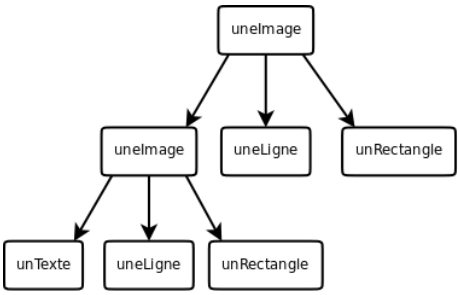


FIG. 5.3: Exemple d'un Objet composite (adapté de [GHJV95]).

Application : on peut utiliser le patron objet composite dans 2 cas :

1. Pour représenter une hiérarchie d'objets avec leurs composants.
2. Lorsque les clients peuvent se permettre d'ignorer la différence entre la composition d'objets et les objets individuels. Les clients traiteront tous les objets de la structure uniformément.

Structure : la structure est illustrée par son diagramme de classes à la figure 5.4. La figure 5.5 propose le diagramme de classes correspondant à l'objet composite de la figure 5.3.

Participants : entre parenthèses se trouve le nom des classes relativement à l'exemple de la figure 5.5.

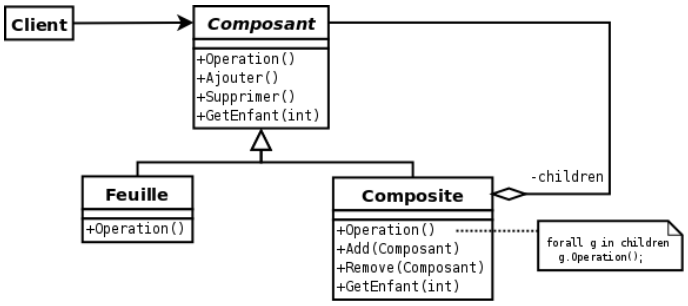


FIG. 5.4: Diagramme de classes du patron de conception Objet composite (adapté de [GHJV95]). Composant est une interface représentant soit un objet simple (une feuille), soit un objet composé (un nœud).

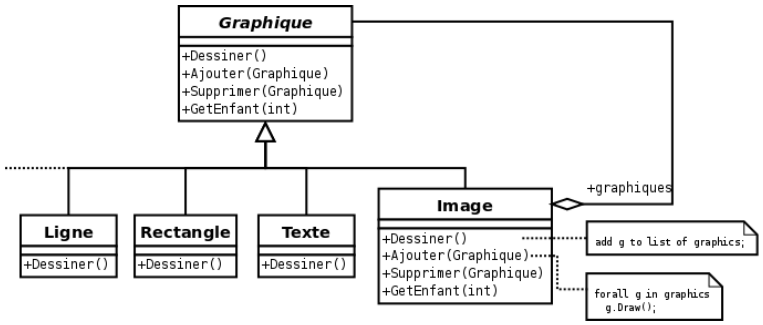


FIG. 5.5: Diagramme de classes d'un exemple d'objet composite (adapté de [GHJV95]).

Composant (Graphique) : définit l'interface pour les objets de la composition ; implémente les comportements par défaut communs à toutes les classes ; déclare l'interface d'accès et de gestion des composants enfants ; définit une interface d'accès au composant parent si nécessaire (facultatif).

Feuille (Rectangle, Ligne, Texte) : représente les objets feuilles de la composition (les feuilles n'ont pas d'enfants) ; définit le comportement des objets primitifs de la composition.

Composite (Image) : définit le comportement des composants ayant des enfants ; conserve les composants enfants ; implémente les opérations liées aux enfants de l'interface Composant.

Client : manipule les objets de la composition via l'interface Composant.

Collaboration : les clients utilisent l'interface Composant pour interagir avec les objets de la structure. S'il s'agit d'une feuille, la requête est gérée directement. S'il s'agit d'un composite, il relaie généralement la requête à ses enfants, avec parfois certaines opérations avant ou après le relais si nécessaire.

Conséquences :

1. *Définit une hiérachie de classes composées d'objets primitifs et d'objets composés.* Les objets primitifs peuvent être composés en objets plus complexes, qui à leur tour peuvent être composés en objets encore plus complexes et ainsi de suite récursivement. Partout où le code du client s'attend à un objet primitif, il peut également prendre en charge un objet composé.
2. *Simplifie le client.* Le client peut traiter un objet primitif et une structure composée uniformément. Les clients ne savent même pas s'ils traitent avec une feuille ou un composite, ce qui simplifie leur code.

3. *Simplifie l'ajout de nouveaux types de composants.* Des composites fraîchement définis ou de nouvelles feuilles fonctionnent automatiquement avec les structures existantes et leur code. Les clients ne doivent donc pas être modifiés.
4. *Généralise parfois trop la conception.* Rendre l'ajout de nouveaux composants aisé a le désavantage qu'il est plus difficile de restreindre les compositions possibles. Il arrive de ne vouloir avoir que certains composants dans un objet composite. Il est impossible [2] de se baser sur les types pour ajouter ces contraintes. Il faut donc ajouter des vérifications à l'exécution.

5.5 Patron de conception et génie logiciel empirique

Il est impossible de présenter tous les travaux relatifs aux patrons de conception réalisés en génie logiciel empirique. Nous allons nous concentrer sur certains en nous focalisant sur le point de vue de la maintenance.

Prechelt *et al.* [PUT+01] ont cherché une évidence empirique concernant les bénéfices tirés de l'utilisation des patrons de conception. Ils se sont intéressés plus particulièrement au cas où il existait des solutions plus simples. D'une manière générale, l'utilisation des patrons de conception était bénéfique. Seuls quelques cas faisaient exception. Bien que la plupart de ces effets étaient prévus, ils eurent tout de même quelques surprises. Ils concluent néanmoins que, à moins qu'il n'y ait une bonne raison à l'utilisation de la solution plus simple, il est judicieux d'utiliser le patron de conception. Les exigences étant amenées à être modifiées, la flexibilité des patrons de conception est un atout. Il faut noter que contrairement à l'étude de Wendorff (présentée ci-après), les auteurs se focalisent sur les utilisations appropriées des patrons de conception.

Wendorff [Wen01] reporte les problèmes qu'il a rencontrés en travaillant sur un gros logiciel commercial (1 000 000 LOC). Il adopte uniquement le point de vue de la ré-ingénierie et se concentre sur les patrons de conception trouvés dans le code en production. Il constate que certains patrons sont mal utilisés et se focalise sur la manière de remédier à ces utilisations inappropriées. Il présente plusieurs utilisations de patrons de conception ayant mené à des résultats négatifs et explique pourquoi ces utilisations étaient inappropriées. D'une manière générale, les patrons de conception affectent certains attributs de qualité mais au détriment d'autres. La flexibilité, par exemple, n'est pas une flexibilité universelle : un patron de conception apporte la flexibilité à un aspect bien particulier de la conception en fonction du contexte. Le contexte et l'aspect doivent donc être bien cernés afin d'utiliser le patron adapté. Une mauvaise utilisation aura de graves conséquences. Certains patrons étaient mal utilisés mais, étant fortement couplés à d'autres parties du code et donnant ainsi lieu à des dépendances complexes, leur retrait n'était pas viable non plus. Il conclut sur la nécessité de développer un rapport coûts/bénéfices plus objectif permettant une meilleure décision d'application ou non d'un patron. De même, il faut des procédures permettant le retrait de patrons de conception devenus obsolètes ou qui ont été mal utilisés.

Prechelt *et al.* [PULPT02] ont tenté de mettre en évidence les avantages liés à la mise en évidence des patrons de conception dans les commentaires d'un logiciel. Malgré un manque avoué d'évidence dû à la conception de l'expérience, ils concluent qu'en fonction du programme, de la tâche de maintenance, de la connaissance du patron de conception et du personnel, les lignes de commentaires concernant les patrons de conception peuvent réduire considérablement le temps nécessaire à un changement dans le programme ou peuvent aider à améliorer la qualité du changement.

Aversano *et al.* [ACC+07] ont mené une étude empirique concernant l'évolution des patrons de conception dans la vie de logiciels. Ils ont analysé la fréquence et l'importance (nombre de modifications de classes ne faisant pas partie du patron) des modifications liées aux patrons de conception.

[2] Du moins, avec les mécanismes généraux des langages orientés-objets. Un langage particulier peut bien entendu disposer de fonctionnalités permettant cette limitation.

Leur conclusion est que la fréquence et l'importance des changements ne dépendent pas du patron de conception utilisé mais du rôle joué par le patron dans l'application. Les patrons changent soit par leur implémentation, soit par l'ajout de sous-classes soit encore par la modification des méthodes des interfaces. Cette dernière étant la source des changements les plus importants dans les classes clientes. Les résultats suggèrent que les développeurs doivent soigneusement considérer l'usage des patrons de conception devant supporter les fonctionnalités importantes de l'application. Ces patrons seront sujets à de fréquents changements et demanderont une grosse activité de maintenance qui sera fortement affectée par un mauvais choix de patron.

Di Penta *et al.* [DPCGA08] ont étudié 12 patrons de conception pour déterminer si les rôles joués par une classe ont une influence sur son taux de modification et si certains types de changement affectent plus certains rôles que d'autres. D'une manière générale, leurs résultats suggèrent de concevoir soigneusement les rôles les plus sujets aux changements. En effet, leur propension aux changements peut rendre les autres parties du systèmes moins robustes aux changements. Cela confirme les résultats d'Aversano *et al.* [ACC+07].

Chapitre 6

Visualisation graphique et oculométrie

Ce chapitre aborde avec la section 6.1 la question des représentations visuelles des patrons de conception. La section 6.2 quant à elle présente plus en détail la technique de l'oculométrie utilisée pour notre expérience.

6.1 Visualisation de patron de conception

Les représentations visuelles font partie du quotidien du programmeur. Elles sont utilisées pour la documentation des logiciels. Les *stimuli*, en oculométrie, sont les objets visualisés par les sujets. Ceux utilisés pour notre expérience sont des représentations visuelles de morceaux de logiciels sous forme de diagrammes de classes UML. La section 6.1.1 explique brièvement l'intérêt des représentations visuelles des logiciels. Dans la section 6.1.2, nous nous intéressons plus particulièrement aux représentations visuelles des patrons de conception. La section 6.1.3 s'attarde sur l'agencement des éléments dans les diagrammes de classes UML.

6.1.1 Intérêt des représentations visuelles

Le but de la compréhension d'un logiciel est d'acquérir suffisamment de connaissances à propos d'un système afin de pouvoir le faire évoluer d'une manière disciplinée. Il faut comprendre les entités, découvrir les relations et créer des abstractions. Ce processus implique l'identification, la manipulation et l'exploration d'artéfacts dans une représentation particulière. Le programmeur doit alors reconnaître des patrons mentaux et assembler tous ces artéfacts pour former une représentation plus abstraite du système [TH03]. La compréhension de logiciel revient donc à se construire une représentation mentale du système sur lequel on travaille [KKB+98]. Pour cela, le programmeur utilise la documentation disponible. Il existe principalement 2 catégories de documents utilisés comme aide à la compréhension de logiciel : la documentation textuelle et la documentation graphique. La documentation graphique peut être subdivisée en 3 types : statique, interactive et éditable [TH03]. Les représentations visuelles sont particulièrement indiquées pour la compréhension de logiciel, l'information pouvant être transmise d'une manière correspondant mieux à notre propre représentation mentale, cela facilite la création du modèle mental utile. Les représentations visuelles permettent en effet de diminuer l'effort de compréhension et d'apprentissage par l'omission des détails non pertinents tout en mettant en avant les informations les plus utiles concernant les objets et leurs relations [CK05] (dans cet article l'auteur parle d'objets et relations de la vie courante et non pas de programmation orientée-objet).

Le lecteur non convaincu peut répondre à la question [1] suivante :

> *Soit la liste d'arêtes suivante, combien y a-t-il d'étapes pour aller du nœud a au nœud b ?*
> (<a,c>,<a,e>,<b,d>,<c,e>,<e,d>)

Maintenant ceci :

> *Soit le diagramme de la figure 6.1, combien y a-t-il d'étapes pour aller du nœud a au nœud b ?*

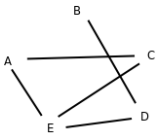


FIG. 6.1: Illustration de l'intérêt des représentations visuelles.

Comme on peut le constater, la réponse à la question est beaucoup plus simple lorsque l'on dispose du schéma. La représentation visuelle permet de présenter les éléments de manière à ce que celui qui la regarde voie l'élément important.

6.1.2 Représentation graphique des patrons de conception

Les représentations graphiques et les patrons de conception étant reconnus comme utiles au programmeur, la question de la représentation graphique des patrons de conception a été étudiée. La connaissance des patrons de conception couplée à leur représentation explicite peut dès lors améliorer la compréhension du fonctionnement d'un logiciel. Inversement, le manque d'informations concernant les patrons peut s'avérer être une lacune pour la compréhension du fonctionnement du logiciel [Por08].

La représentation standard des patrons de conception sont les diagrammes de classes et les diagrammes de collaboration UML [RJB04]. Mais certaines lacunes dans les diagrammes de classes ont poussé quelques auteurs à proposer d'autres représentations [Por08]. Un des premiers travaux [Vli98] concernant la représentation des patrons de conception a été réalisé par Vlissides, un membre du GoF. Il y déclare se demander pourquoi le GoF n'y a pas pensé lors de la rédaction de leur ouvrage. Il y explique également que les conventions dans le code, telles qu'ajouter le nom du patron dans le nom de la classe concernée, n'est pas viable dans les cas où la classe participe à plusieurs patrons : « AbstractFlyweightFactorySingleton » n'est pas le pire exemple qu'il ait pu voir. C'est pourquoi, il propose une critique de 3 notations tirées de la littérature.

La littérature propose plusieurs alternatives de présentation des patrons de conception. Ces alternatives vont de fortement visuelles [SK98] à fortement textuelles [DYZ07]. Ces représentations peuvent être divisées de 2 manières différentes [Por08] :

- Par rapport à UML :
  - Basées sur UML : [LK98], [SK98], [FKGS04], [TT07] et [DYZ07].
  - Non basées sur UML : [Gam98], [EYG97] et [MHG02].
- Par rapport au nombre de diagrammes :
  - Mono-diagrammes : [Gam98], [SK98], [TT07] et [DYZ07].
  - Multi-diagrammes : [LK98] et [FKGS04].

D'une manière générale, une représentation graphique de patrons de conception doit trouver un juste milieu entre plusieurs points importants :

- Le rôle de chaque classe doit être explicite.

[1] Tirée de [Nic94]

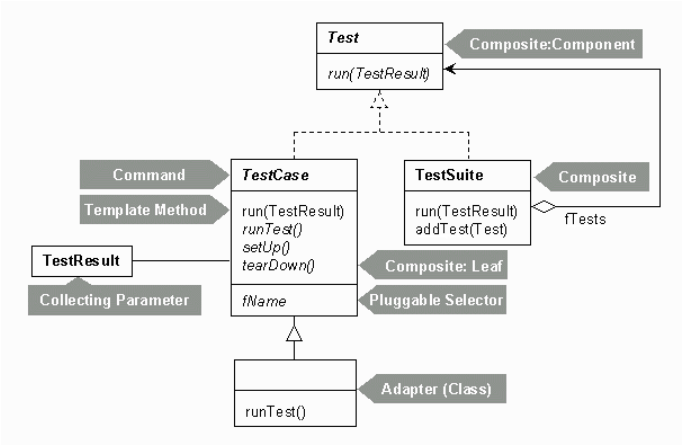


FIG. 6.2: Illustration de la représentation « annotations patron : rôle » (repris de [KB09]).

- Les notations ajoutées au diagramme ne doivent pas nuire à sa lisibilité lorsque la taille est conséquente ou lorsque que les classes participent à plusieurs patrons.
- La représentation doit être imprimable à faible résolution (*e.g.*, une mauvaise combinaison de couleurs).
- La structure des patrons n'est pas suffisante à leur identification (*e.g.*, État et stratégie ont la même structure).

A titre d'exemple, la figure 6.2 illustre une représentation, appelée « annotations patron : rôle », proposée par GAMMA [Gam98] et critiquée dans [Vli98]. Dans cette représentation, une boîte foncée est ajoutée avec le nom du patron et le(s) nom(s) du (des) participant(s) associé(s) à la classe. Cette représentation ne cause aucune interférence avec le diagramme et favorise la lisibilité. Malheureusement, elle résiste mal à une augmentation de la taille des diagrammes et l'impression peut donner des résultats difficiles à lire [Por08].

À notre connaissance, la seule étude empirique réalisée concernant ces différentes notations est celle de Porras [Por08]. Il y réalise une étude à l'aide d'un oculomètre afin de comparer les performances de sujets effectuant des tâches basiques en relation avec des patrons de conception. Le but est de mesurer l'effet induit par les représentations [SK98], [DYZ07] et [Gam98] par rapport à la représentation standard UML. Sa conclusion est que pour certaines tâches, la représentation proposée par DONG *et al.* [DYZ07] semble légèrement plus efficace que la représentation standard.

6.1.3 Agencement de graphiques

L'UML [RJB04] est la représentation graphique qui s'est imposée *de facto*. Mais en tant que standard de représentation, l'UML ne dit rien de la manière de produire des diagrammes lisibles. Un accord doit donc être trouvé, particulièrement pour le partage de grands diagrammes, concernant l'esthétique afin de réduire les coûts de communication et de minimiser les incompréhensions provenant d'agencements différents d'un même diagramme [Eic03]. Malgré cette nécessité reconnue, les outils ne produisent pas automatiquement de diagrammes avec un agencement acceptable et souvent ne respectent même pas la dernière norme UML en vigueur [HE03].

Des différents types de diagrammes définis par le standard, seuls les diagrammes de classes semblent avoir fait l'objet de beaucoup de recherches. Le lecteur peut consulter [PMCC01] ou le récent [Eic08] qui sont des contre-exemples avec les diagrammes de collaboration et les diagrammes de cas d'utilisation. Les premiers travaux se sont concentrés sur les graphiques en général et non sur les diagrammes de classes UML [WPCM02]. HERMAN *et al.* [HMM00], par exemple, se sont intéressés à la visualisation des graphes ainsi qu'à la navigation dans ceux-ci. PURCHASE [Pur97]

a effectué une expérience visant à étudier les effets de 5 caractéristiques esthétiques des graphes.

Eichelberger, [Eic02] et [Eic03], a introduit un ensemble de critères d'esthétique propres aux diagrammes de classes UML. Il propose également un prototype d'algorithmes d'agencement automatique. Bien que ce problème d'agencement automatique de diagramme soit un problème NP-complet, il existe des approches qui semblent prometteuses. Il propose un ensemble de critères à prendre en compte pour l'agencement de diagrammes de classes sur base de la modélisation du logiciel représenté et sur base de ses critères d'esthétique. Ces critères prennent donc également en compte la structure sémantique des fonctionnalités. Il y propose également une liste (incomplète selon lui) d'indicateurs faisant office d'alerte à problèmes de modélisation du logiciel. Gutwenger *et al.* [GJK+03] proposent une approche pour visualiser les diagrammes de classes UML se basant sur un mixte des critères esthétiques de [Pur97] combinés avec des critères de perception visuelle.

Andriyevska *et al.* [ADSM05] pensent qu'il est plus important de prendre en compte l'importance architecturale des classes dans le diagramme. Ils ont mené une étude pilote qui semble montrer qu'un agencement construit sur cette base aide plus qu'un agencement construit sur base de critères d'esthétisme. Selon eux, un diagramme devrait donc exprimer d'abord l'architecture.

Purchase *et al.*, [PMCC01],[PCM+01], ont mené diverses études empiriques concernant les arrangements automatiques de diagrammes de classes. Dans [PCM+01], les auteurs définissent les règles de 2 variantes d'arrangement automatique. Dans [PMCC01], les auteurs tentent d'identifier les critères esthétiques les plus importants du point de vue de la compréhension humaine.

Dwyer [Dwy01] discute l'utilisation d'une représentation en 3 dimensions de diagrammes de classes UML comme solution pour les diagrammes de grande taille. Suite à une étude pilote, il conclut, sous réserve d'études plus approfondies, que sa méthode améliore la connaissance de la structure des modèles de systèmes complexes.

De cette analyse de la littérature, on se rend compte qu'il existe de nombreux critères et que certains entrent en conflit avec d'autres. C'est pourquoi, souvent, la confusion règne pour les outils lorsqu'il faut choisir le critère approprié [SW05]. L'inefficacité des outils dans l'arrangement automatique des diagrammes de classes UML pointée par [HE03] n'est donc pas anodine. Nous y voyons le symptôme d'un domaine encore immature.

Plus particulièrement, on peut se poser la question de l'agencement des patrons de conception. Généralement, 2 principes sont retenus. Soit on utilise la *forme canonique* du patron qui est celle proposée par Gamma *et al.* dans leur ouvrage, soit on utilise une autre forme [Por08]. La forme canonique n'est cependant pas toujours utilisable dans la mesure où la structure d'une implémentation d'un patron de conception peut s'écarter de la structure de base définie dans [GHJV95].

6.2 Oculométrie

L'*oculométrie* est une technique où les mouvements individuels des yeux sont enregistrés. Le chercheur et l'observateur connaissent tous deux la séquence d'analyse d'une scène visuelle de l'observateur. Lorsque l'on s'intéresse aux techniques d'oculométrie, la première question à se poser est celle de la motivation de l'enregistrement des mouvements oculaires. Nous bougeons nos yeux pour amener une zone particulière de la rétine (celle qui permet la vision à haute résolution : la fovea) à voir en détail ce qui se trouve au centre du regard. Nous plaçons notre attention sur ce point de manière à nous concentrer (au moins pour un bref instant) sur l'objet ou la région d'intérêt. Nous pouvons donc supposer que, si nous traçons les mouvements des yeux de quelqu'un, nous pouvons suivre le cheminement de son attention. Cela peut nous donner un aperçu de ce que l'observateur trouve intéressant, ce qui attire son attention et peut-être même fournir une indication de la manière dont il perçoit la scène qu'il regarde [Duc07].

La section 6.2.1 explique l'intérêt des représentations visuelles. La section 6.2.2 explique en quoi un oculomètre est utile. La section 6.2.3 explique les différents types de mouvements oculaires et la

section 6.2.4 présente différentes mesures que l'on peut en déduire. Enfin, la section 6.2.5 propose un petit état de l'art de l'utilisation de l'oculométrie en génie logiciel.

6.2.1 Intérêt de l'attention visuelle

Il est intéressant de se demander ce qu'est l'attention visuelle et si les mouvements des yeux reflètent bien quelque chose en rapport avec le processus cognitif de l'attention visuelle. L'humain est un être fini et ne peut faire attention à toutes les choses en même temps. Généralement, l'attention est utilisée pour focaliser notre capacité mentale sur certaines sensations de manière à ce que l'esprit puisse prendre en compte correctement le *stimulus* qui l'intéresse. Le *stimulus* est défini en oculométrie comme l'objet visualisé par le sujet. Notre capacité de traitement de l'information est limitée. Le cerveau traite les sensations en se concentrant sur des composantes spécifiques du domaine sensoriel de sorte que les visions intéressantes, les sons intéressants, les odeurs intéressantes ou autres soient examinés avec plus d'attention aux détails que les *stimuli* périphériques. C'est particulièrement vrai pour la vision. La vision humaine est un processus par à-coups se basant sur l'intégration de la perception de petites régions pour construire une représentation cohérente de l'ensemble [Duc07].

6.2.2 Utilité de l'oculomètre

L'attention visuelle semble donc refléter « la dernière couche » du processus cognitif. L'hypothèse « œil-esprit » signifie que l'enregistrement des mouvements oculaires peut fournir une trace de l'attention visuelle d'une personne en rapport avec un affichage [PB04].

Bien que l'étude de l'attention visuelle semble prometteuse, il faut se demander si l'utilisation d'un *oculomètre* est indispensable. Un oculomètre est un appareil enregistrant d'une manière ou d'une autre le parcours oculaire d'une personne. Il reste encore aujourd'hui fort cher et son utilisation n'est pas toujours facile. La calibration est un processus qui peut devenir un cauchemar lorsque les sujets ont des lunettes, des lentilles de contact, du maquillage ou des yeux trop foncés. La précision risque alors de ne pas être au rendez-vous et la validité de l'expérience en sera affectée. Bien que ces barrières tendent à disparaître grâce à l'évolution de la technique, certains ont cherché des alternatives au matériel cher (surtout s'il est accompagné d'un logiciel d'analyse qui, lui aussi, est facturé) [JH06].

Johansen et Hansen [JH06] se sont posé la question et ont étudié 2 alternatives bon marché. La première se base sur le fait qu'une personne a tendance à parcourir de la même manière quelque chose qu'elle a déjà vu précédemment. La seconde se base sur les prédictions d'un expert. Leur conclusion est que la première méthode devrait donner des indications fiables, bien que pas nécessairement à 100%. Cela ne devrait pas mener à des conclusions fausses basées sur des données biaisées mais les limitations possibles des méthodes doivent être claires pour tous ceux qui les utilisent. Leur expérience indique qu'il y a des limitations sévères aux souvenirs des personnes concernant la vue de certains éléments tels que les logos. Bien que Babcock *et al.* [BPF03] rejettent totalement la fiabilité des données reportées par l'observateur lui-même, les auteurs de cette expérience-ci trouvent qu'il est intéressant de continuer à améliorer les méthodes d'auto-reportage et d'identifier les meilleures conditions pour leur utilisation. La plus grosse limite de cette méthode est qu'elle impose des tâches courtes et simples, la mémorisation de longues séquences étant plus propice à l'erreur. La seconde méthode, par contre, s'est montrée beaucoup moins fiable.

Il semble donc que la méthode d'auto-reportage ne soit pas dénuée d'intérêt malgré les améliorations techniques des oculomètres. Certains domaines nécessitant des études à grande échelle et ne nécessitant pas d'analyses trop longues des *stimuli* peuvent en effet profiter de la facilité de mise en œuvre de ce type de technique [BPF03].

Au-delà des études telles que celles proposées dans ce document, il faut envisager des applications où l'oculomètre est utilisé comme périphérique d'entrée permettant une interaction avec une interface. Un utilisateur pourrait alors positionner le curseur simplement en regardant l'endroit où il veut aller [PB04].

6.2.3 Types de mouvements oculaires

L'*acuité visuelle* est la « faculté » qui permet de distinguer nettement, de voir séparément 2 points, 2 lignes, 2 objets. Le *champ visuel* est la portion de l'espace dans les limites de laquelle la rétine d'un œil immobile perçoit des rayons lumineux. Le centre de ce champ ou point de fixation est situé sur l'axe central du regard. Le champ visuel est limité en haut, en bas, du côté nasal et temporal, par les points les plus périphériques d'où les rayons lumineux impressionnent la rétine. Le champ visuel est limité comme suit : 90° du côté temporal, 55° du côté nasal, 50° en haut, 60° en bas. La *fovea* est la zone avec la meilleure acuité visuelle [AM72].

Le positionnement de la fovea se fait principalement grâce à 5 mouvements de base. Les mouvements vestibulo-oculaires et optocinétiques permettent de fixer le regard sur une cible lorsque la tête bouge, ce sont des mouvements de stabilisation de l'image. Les saccades, la poursuite lente et les mouvements de vergence ont, à l'opposé, pour fonction de garder le regard sur une cible même lorsque celle-ci se déplace ; ce sont des mouvements d'orientation du regard [RT06].

- Mouvements vestibulo-oculaires : ils maintiennent les images stables sur la rétine durant les rotations brèves et rapides de la tête.
- Mouvements optocinétiques : ils maintiennent les images stables sur la rétine durant des rotations lentes et prolongées de la tête.
- Saccades : elles amènent l'image d'un nouvel objet sur la fovea ; le mouvement rapide de l'œil est une saccade.
- La poursuite lente : elle maintient l'image d'un objet en mouvement sur la fovea.
- Mouvements de vergence : ils ajustent l'orientation des yeux en fonction de la distance de l'objet en profondeur ; ils permettent la perception de la perspective.

Nous allons voir plus en détail les mouvements d'orientation du regard : saccades et poursuite lente. Nous développerons aussi la notion de fixation qui reprend *grosso modo* les effets des mouvements de stabilisation de l'image.

Saccades

Les *saccades* sont des mouvements rapides utilisés pour repositionner la fovea à un nouvel endroit de l'environnement visuel. Elles sont volontaires et réflexives. Volontaires lorsque l'observateur positionne son regard ailleurs. Réflexives lorsqu'elles sont effectuées pour ramener sous la fovea une cible qui a bougé soudainement. On pense que la destination d'une saccade est pré-programmée. Une fois que la localisation de l'endroit de fixation suivant est calculée, la saccade ne peut être altérée. Une des raisons de cette supposition est que, comme la saccade dure de 10 ms à 100 ms, il n'y a pas le temps nécessaire pour un retour d'information capable de guider l'œil. On parle de 200 ms pour la programmation. D'un autre côté, un système de retour d'information est possible dans le cas où des copies internes de la tête, de l'œil et de la destination sont utilisées pour guider l'œil durant la saccade.

La saccade est le mouvement le plus rapide de l'œil. Elle peut atteindre 900°/s. La forme d'une saccade est très standard : une accélération progressive vers un pic de vitesse et une décélération. La vitesse maximale et la durée de la saccade dépendent de l'amplitude de la saccade. Cette relation est très stable pour un individu mais est altérée par la drogue, l'alcool et la fatigue [RT06] et [Duc07].

Poursuite lente

Les mouvements de poursuite se présentent lorsqu'il faut suivre un objet en mouvement. Les yeux se déplacent dans l'espace pour maintenir l'image de l'objet fixe. L'œil est capable de s'adapter à la vitesse de l'objet. Ce mouvement est volontaire mais nécessite une cible. La vitesse est linéaire à partir de 40°/s et va jusqu'à environ 100°/s. La drogue, l'alcool et la fatigue dégradent la qualité de la poursuite lente.

Lors de l'arrivée d'un objet en mouvement, l'œil effectue une saccade pour positionner la fovea sur la cible, ensuite l'œil suit la cible de manière continue. La latence du système est d'environ 100 à 130 ms [RT06] et [Duc07].

Fixation

Les fixations sont la stabilisation de la rétine sur un objet. Elles sont caractérisées par des mouvements miniatures des yeux : tremblements, dérives et micro-saccades. Une image complètement stabilisée par rapport à la rétine n'est pas perçue plus d'une seconde. Ce sont ces micro-mouvements qui permettent le rafraîchissement de l'image. Ces micro-mouvements qui caractérisent effectivement les fixations sont considérés comme du bruit pour les systèmes d'oculométrie. Ce bruit apparaît aléatoirement dans un périmètre d'environ 5° de l'angle visuel. Bien que cette considération comme du bruit simplifie fortement le processus cognitif, cela permet la simplification de la modélisation du signal. Enfin, il faut noter que les fixations représentent approximativement 90% du temps de regard. [Duc07].

L'analyse des mouvements oculaires

Les 3 mouvements décrits sont des mouvements survenant de manière volontaire ou réflexive. Bien que l'on n'exclue pas leur survenance de manière involontaire, on fait l'hypothèse que ces mouvements fournissent suffisamment de preuve de la volonté de l'attention visuelle. Les fixations correspondent naturellement au désir de maintenir le regard sur un objet source d'intérêt. Il en va de même pour la poursuite d'un objet et les saccades sont considérées comme un désir volontaire de changer l'objet de l'attention [Duc07].

6.2.4 Mesures en oculométrie

Pour rappel (voir section 3.1.4), il existe 2 types de mesures : les mesures directes et les mesures indirectes, qui sont dérivées des mesures directes. Nous discutons dans la suite des mesures directes généralement envisagées et des principales mesures indirectes les plus populaires en oculométrie.

Mesures directes

Les mesures directes généralement utilisées ne sont pas nombreuses. Il s'agit des saccades, des fixations, du parcours oculaire, de la taille de la pupille et du taux de clignement des yeux.

Saccade : aucune perception ne se passe durant les saccades. Elles ne peuvent donc donner aucune information concernant la complexité ou l'importance d'un objet de la scène.

Fixation : l'interprétation des fixations peut varier avec le contexte. Dans une tâche d'encodage, un haut taux de fixation sur une zone particulière peut indiquer un plus grand intérêt de la zone ou une plus grosse difficulté d'encodage. Lors d'une tâche de recherche par contre, un grand nombre de fixations ou un groupe de fixations sur une zone sont le signe d'une plus grande incertitude concernant la reconnaissance de la cible. Le temps de la fixation est

aussi lié au temps de traitement de l'objet fixé. Il est largement accepté qu'une représentation associée à de longues fixations n'est pas aussi significative pour l'observateur qu'une représentation associée à de courtes fixations [PB04].

Parcours oculaire : il décrit une séquence complexe de « fixation-saccade-fixation-etc. ». Pour une tâche de recherche, le chemin oculaire optimal est une ligne droite vers la cible désirée, avec de relatives courtes fixations sur la cible [PB04].

Taille de la pupille et clignement des yeux : la taille de la pupille et le taux de clignement des yeux peuvent être un indicateur de la charge de travail cognitive. Un faible taux de clignement des yeux indique une charge cognitive plus importante et un taux plus élevé peut indiquer la fatigue. De larges pupilles peuvent aussi indiquer un effort cognitif plus important. Mais ces 2 facteurs peuvent aussi être influencés par beaucoup d'autres facteurs tels que la lumière ambiante. Ils sont donc peu utilisés [PB04].

Difficultés techniques d'identification des fixations

La nature du regard pose des difficultés techniques pouvant avoir un impact sur les résultats d'une étude. Les 2 mesures principalement utilisées sont les fixations et les saccades. Leur analyse nécessite leur séparation d'une manière ou d'une autre. On traduit généralement le mouvement de l'œil uniquement en emplacements de fixation avec une saccade implicite entre 2 fixations consécutives. Cette manière de procéder réduit considérablement la taille et la complexité du mouvement oculaire. En effet, on retire les saccades brutes de l'analyse et on agrège les mouvements de stabilisation de la rétine en un point représentatif de la fixation. Cette simplification est utile pour 2 raisons. La première a déjà été énoncée et vient du fait qu'aucun processus d'interprétation ne se déroule durant les saccades. Elles sont donc relativement inutiles pour beaucoup d'études. La deuxième est que les micro-mouvements se produisant durant une fixation sont aussi généralement inutiles pour la plupart des études. La réduction à l'identification des fixations diminue donc la complexité, tout en conservant les caractéristiques essentielles à la compréhension des comportements visuels et cognitifs. Mais, s'il est généralement admis que la cognition visuelle ne se passe que durant les fixations, il est plus difficile de savoir exactement quand une fixation commence et se termine. Cette identification est souvent un aspect critique de l'analyse des données et peut avoir un impact significatif sur les résultats [SG00].

Shic *et al.* [SSC08] ont évalué plusieurs algorithmes populaires pour l'identification des fixations et saccades. Ils montrent qu'en modifiant les paramètres des algorithmes, ils changent les durées de fixation d'une manière systématique. Néanmoins, ce systématisme et le comportement cohérant des algorithmes permet la convertibilité de durées moyennes de fixation différentes. Ils laissent d'ailleurs ouverte la question du choix du meilleur algorithme et concluent sur le fait que la question des paramètres des algorithmes n'est pas pertinente. La vraie question est de savoir quel est le lien entre le changement de paramètres et la mesure des résultats.

Mesures indirectes

Les difficultés de liaison des mesures à l'activité cognitive sont probablement la plus grosse barrière à une plus grande utilisation des oculomètres [JKD02]. Les mesures indirectes peuvent être classées selon leur origine : dérivées des saccades, dérivées des fixations ou dérivées du parcours oculaire [PB04]. Elles peuvent également être classifiées selon leur signification : mesure de l'effort cognitif ou mesure de l'effort de recherche [GK99]. Étant donné qu'il n'existe pas encore de concensus concernant la signification de toutes les mesures indirectes et que leur signification est dépendante du contexte, nous préférons utiliser la première classification.

Avant de présenter les mesures indirectes, il nous faut introduire le concept de zone d'intérêt :

Zone d'intérêt (ZI) : zone d'un environnement visuel qui intéresse les chercheurs ou l'équipe de conception. Elle n'est donc pas définie par les participants [JKD02]. Dans le cadre de diagrammes de classes UML, il s'agit généralement des classes et relations.

Zone d'intérêt pertinente (ZIP) : sous-ensemble de l'ensemble des zones d'intérêt revêtant un intérêt plus particulier que les autres. Dans le cadre d'une étude impliquant des diagrammes de classes, il s'agit d'une (ou plusieurs) classe(s) nécessaire(s) à l'élaboration de la réponse à la tâche.

Cible : cette notion est surtout pertinente dans les études concernant les interfaces Homme–Machine. Il s'agit de l'élément du *stimulus* que le sujet doit trouver pour remplir la tâche qui lui a été assignée.

[GK99], [JKD02] ou [PB04] proposent un recueil de mesures dont nous nous sommes largement inspirés. Il faut noter que ces mesures ont été utilisées dans le cadre des interfaces Homme–Machine. Leur signification n'est de ce fait peut-être pas la même dans le cadre de la consultation de diagrammes de classes UML. Il faut donc, lors de la planification d'une expérience en oculométrie, prendre soin d'étudier la signification de la variable dans le contexte d'utilisation. Nous reviendrons sur ce sujet dans la section 7.2.3, lors de la sélection des variables dépendantes de notre expérience.

Mesures dérivées des fixations

1. Nombre global de fixations : plus il est grand, plus la recherche est inefficace.
2. Fixations par zone d'intérêt : un nombre plus important de fixations dans une zone particulière indique soit qu'elle est plus compliquée à comprendre, soit qu'elle est plus importante pour l'observateur.
3. Temps de regard par zone d'intérêt : la proportion de temps passé dans une zone d'intérêt particulière. Cette mesure ne doit pas être confondue avec le nombre de fixations, qui reflète la fréquence et donc l'importance. Le temps passé reflète la difficulté d'interprétation de l'information.
4. Temps de regard moyen dans chaque zone d'intérêt : cette mesure reflète aussi la difficulté d'extraction ou d'interprétation de l'information.
5. Fixations par zone d'intérêt, ajustées par la longueur du texte : si les zones d'intérêt ne sont composées que de texte, le nombre moyen de fixations par zone d'intérêt devrait être divisé par le nombre moyen de mots du texte. Cela permet de lisser le résultat : un nombre plus important de fixations simplement dû à un nombre plus important de mots ; un plus grand nombre de fixations à cause d'une entité plus difficile à reconnaître.
6. Durée de fixation : une plus grande durée de fixation indique soit une plus grande difficulté dans l'extraction de l'information, soit que l'objet est plus attrayant pour une raison ou une autre.
7. Regard : le regard est habituellement la somme des durées de fixation dans une zone prescrite. Cette mesure permet de comparer l'attention distribuée entre les cibles.
8. Densité spatiale des fixations : des fixations concentrées dans de petites zones indiquent une recherche efficace et ciblée. Des fixations fortement distribuées indiquent une recherche inefficace.
9. Fixation post-cible : un grand nombre de fixations hors de la cible, après qu'elle ait été fixée, indique son manque de visibilité ou d'évidence.
10. Temps jusqu'à la première fixation sur la cible : un temps plus court jusqu'à la première fixation sur un objet ou une zone signifie qu'il attire plus l'attention.
11. Pourcentage de participants fixant une zone d'intérêt : si une faible proportion des participants fixe une zone importante pour la tâche, elle devrait peut-être être mise en évidence ou déplacée.

12. Rapport de fixation (on-target all) : le nombre de fixations sur la cible divisé par le nombre de fixations totales est un indicateur de l'efficacité de la recherche. Un faible rapport indique une mauvaise efficacité.
13. Rapport de fixation (in-zip in-zi) : le rapport entre le nombre de fixations dans les zones d'intérêt pertinentes et le nombre de fixations dans les zones d'intérêt en général. Cette mesure permet de lisser les résultats en excluant les fixations dans les zones vides. Ces fixations ne sont en effet pas faciles à interpréter. Ce rapport est un indicateur de la pertinence de l'effort.
14. Taux normalisé de fixation pertinente : cette mesure est proposée par [Jea08]. Il s'agit du rappot entre le nombre normalisé de fixations pertinentes (nombre de fixations dans les ZIP divisé par le nombre de ZIP) et le nombre normalisé de fixations non pertinentes (nombre de fixations ZI divisé par le nombre de ZI). Cette mesure représenterait l'effort d'un sujet. Nous y voyons plus une mesure de la *pertinence* de l'effort. Néanmoins, plus l'effort d'un sujet est pertinent, moins il fait d'efforts.
15. Nombre de fixations (in)volontaires : on peut considérer les fixations courtes inférieures à 240ms comme involontaires et les fixations longues supérieures à 320ms comme volontaires. Cette méthode doit être vérifiée.

Mesures dérivées des saccades

1. Nombre de saccades : plus de saccades indique plus de recherche.
2. Amplitude de saccade : de larges amplitudes de saccades sont le signe de repères significatifs puisque l'attention est redirigée.
3. Saccades régressives : les saccades régressives sont les saccades qui reviennent en arrière. Les régressions indiquent la présence de repères peu significatifs.
4. Saccades révelant un fort changement de direction : chaque saccade de plus de 90° par rapport à la précédente montre un changement rapide de direction. Cela peut signifier que l'utilisateur a changé d'objectif ou que l'information n'a pas répondu à ses attentes.

Mesures dérivées du parcours oculaire

1. Temps de parcours : une plus longue durée indique une recherche moins efficace. Yusuf *et al.* [YKM07] pensent que dans le cas de la lecture, cette mesure est inexacte car les sujets ont des vitesses de lecture inégales.
2. Longueur du parcours : une plus grande longueur indique une recherche moins efficace ou une grande dispersion des éléments.
3. Densité spatiale : une densité spatiale plus petite indique une recherche plus directe.
4. Matrice de transition : la matrice de transition révèle l'ordre de la recherche en terme de transition d'une zone à une autre. Deux parcours avec la même densité spatiale et le même périmètre convexe peuvent avoir des valeurs de transition très différentes.
5. Direction du parcours : cela peut déterminer la stratégie de recherche d'un participant (de haut en bas, ou de bas en haut, etc.).
6. Rapport saccades/fixations : cela compare le temps passé à chercher (saccades) au temps passé à traiter l'information (fixations). Un plus grand rapport indique plus de traitement et moins de recherche.

6.2.5 L'oculométrie en génie logiciel expérimental

Malgré une utilisation fréquente dans divers domaines ces dernières années, l'utilisation d'oculomètres en génie logiciel n'est que très récente. Ces techniques ont déjà beaucoup été utilisées pour des études d'utilisabilité [PB04] ou d'interface pour le Web par exemple [GSL+02].

Bednarik et Tukiainen [BT04] ont mené une étude afin de comparer 2 outils cherchant à tracer l'attention visuelle : un oculomètre et un visualisateur à focalisation restreinte. Ce second appareil brouille simplement l'ensemble de l'écran sauf une partie qui est celle visionnée par l'observateur. L'étude avait pour but d'étudier les comportements des participants avec un outil de débuggage. Un tel outil fournit différentes représentations dans des zones adjacentes et, pendant la compréhension des programmes, les programmeurs utilisent ces différentes zones pour se construire une représentation mentale. Ils doivent coordonner les représentations. Ils concluent que le visualisateur à focalisation restreinte influence la dynamique du comportement des participants mais ne provoque pas de modifications concernant le temps passé dans les différentes zones.

Toujours Bednarik et Tukiainen [BT06] utilisent un oculomètre afin d'étudier le processus de compréhension d'un logiciel par les programmeurs. Les programmeurs interagissent avec un outil de visualisation. Ils présentent une approche permettant l'analyse des données et appliquent leur méthode pour caractériser les stratégies et comportements que les programmeurs adoptent pour coordonner les représentations multiples d'un programme durant la compréhension. Ils concluent que l'oculométrie peut contribuer à notre compréhension des processus cognitifs impliqués dans la compréhension de programme, le débuggage et la visualisation.

Guéhéneuc [Gué06] a utilisé un oculomètre afin de récolter des données concernant l'utilisation de diagrammes de classes UML durant la compréhension de logiciels par des programmeurs. Il s'intéresse à l'acquisition des informations. Il introduit une technique de visualisation pour rassembler et présenter ses données et propose 2 études de cas. Ces études de cas montrent que les programmeurs commencent par naviguer dans le diagramme de manière plutôt aléatoire afin d'identifier les parties les plus utiles avant de se concentrer sur ces parties. Un résultat plus surprenant est que les programmeurs ne semblent pas suivre les relations binaires entre les classes telles que les compositions et l'héritage.

Yusuf *et al.* [YKM07] a également utilisé un oculomètre afin de collecter des données concernant l'activité de sujets effectuant une tâche donnée sur des diagrammes de classes UML. Ils veulent mieux comprendre la manière qu'ont les programmeurs d'explorer un diagramme de classes UML afin de développer de meilleures techniques d'agencement des classes. Leurs résultats montrent que les experts en UML ont tendance à utiliser les choses telles que les stéréotypes, couleurs et agencements pour naviguer plus efficacement dans les diagrammes. De plus, les experts ont tendance à commencer l'exploration au centre pour ensuite s'en écarter.

Pietinen *et al.* [PBG+08] proposent une méthodologie de recherche dans le cas où l'on étudie l'attention visuelle de 2 programmeurs travaillant par paire sur un seul *simulus*. Ils concluent que la mise en place d'un système capturant les regards de 2 programmeurs durant des tâches réelles de développement logiciel est bien plus compliquée que dans le cas où l'on n'étudie qu'une personne. Selon eux, les oculomètres que l'on pose sur la tête ne remplissent pas les conditions nécessaires à la réalisation d'une telle tâche. La plupart des problèmes rencontrés étaient de nature technique, ce qui leur fait dire que les évolutions futures devraient permettre ce type d'études.

Jeanmart [Jea08] a réalisé une expérience afin de déterminer si le patron de conception « Visiteur » a un impact positif sur la compréhension logicielle. Malheureusement, le manque de consensus sur la signification des mesures l'a empêché d'obtenir un résultat concluant.

Chapitre 7

Conception de l'expérience

La conception de cette expérience suit la méthodologie proposée par [WRH+00] et présentée dans la section 3.2. La section 7.1 définit formellement le but de l'expérience. La section 7.2 explique les différents choix effectués durant la phase de planification. Enfin, la section 7.3 explique le déroulement proprement dit de l'expérience.

7.1 Définition

Nous définissons notre expérience suivant le canevas présenté dans la section 3.2.3 :

> Analyse du (des) <Objet(s) d'étude>
> dans le but de <Objectif >
> selon la <Focalisation sur la qualité>
> sous l'angle de <Perspective>
> dans le contexte du <Contexte>

La définition de l'expérience est la suivante :

Objet d'étude : utilisation des patrons de conception dans des logiciels indépendants. Les patrons de conception considérés sont ceux décrits par le GoF [GHJV95].

Objectif : évaluer l'impact de la connaissance de ces patrons lors d'opérations de maintenance.

Focalisation sur la qualité : effort cognitif, effort de recherche et qualité de l'opération de maintenance.

Perspective : différence entre le développeur expert et le développeur novice.

Contexte : laboratoire de génie logiciel de l'École Polytechnique de Montréal avec des étudiants et du personnel académique comme sujets, ainsi que dans les locaux de l'entreprise Pyxis [1] avec une partie de son personnel comme sujets.

7.2 Planification

7.2.1 Sélection du contexte

Le contexte d'une expérience est défini par les critères suivants :

[1] `http://www.pyxis-tech.com/fr/`

- hors ligne ou en ligne
- étudiant ou professionnel
- problème de petite taille ou problème réel
- spécifique ou général

Les sous-sections suivantes proposent un rapide rappel de la signification de ces critères et expliquent les choix effectués pour la définition du contexte de l'expérience. L'expérience se déroulant dans le cadre d'un stage avec un budget limité, les choix s'imposent souvent d'eux-mêmes.

Hors ligne ou en ligne

Une expérience en ligne étudie l'objet dans les conditions réelles où il apparaît. Une expérience hors ligne va tenter de recréer artificiellement les conditions d'apparition de l'objet étudié. Le plus souvent, ce type d'expérience prend alors place dans un laboratoire.

Une expérience en ligne est plus intéressante qu'une expérience hors ligne mais est plus difficile à contrôler et plus coûteuse. C'est pourquoi l'on ne procède généralement à des expériences en ligne qu'après avoir obtenu des résultats satisfaisants grâce à des expériences hors ligne que l'on va alors tenter de confirmer en ligne [WRH+00].

Notre expérience s'effectue donc hors ligne. Ceci est dû au fait qu'il n'existe à l'heure actuelle que peu de résultats empiriques concernant les divers bénéfices des patrons de conception. De plus, les coûts, notamment en terme de temps (4 mois), pour la réalisation d'une telle expérience ne sont pas compatibles avec les moyens alloués pour le stage.

Étudiant ou professionnel

Deux catégories de sujets sont généralement considérées lors d'une expérience en génie logiciel : des étudiants (1^{er}, 2^{e} et 3^{e}cycle) ou des professionnels. Il est difficile pour un chercheur d'obtenir des sujets venant du monde professionnel. L'employeur n'est pas toujours disposé à payer ses employés pour du temps qu'ils ne passent pas pour l'entreprise. Passer l'expérience sur le temps libre d'un employé n'est pas envisageable si l'on désire en recruter un nombre assez important. L'avantage des étudiants est qu'ils sont sur place mais ils ont le désavantage de ne pas toujours être représentatifs de la population que le chercheur désire étudier.

Ont participé à notre expérience : des étudiants des 3 cycles, 8 employés de la société Pyxis, qui a autorisé l'entrée du matériel dans ses locaux durant quelques jours, et 1 professionnel durant son temps libre. Nous ne pouvons donc pas positionner notre expérience par rapport à ce critère.

Problème de taille réduite ou problème réel

Le matériel utilisé oblige l'utilisation de diagrammes pouvant tenir sur un écran 19". La taille d'un diagramme d'application réelle étant beaucoup plus grande que celle permise par la configuration physique, le problème est réduit et même s'il est tiré d'une application réelle, nous le caractérisons de réduit. Il est à noter que la taille en nombre de classes, méthodes, etc. des diagrammes est également dictée par d'autres contraintes telles la charge cognitive.

Spécifique ou général

Le but de toute expérience est de tirer des conclusions à propos d'un objet (section 7.1). Avant de commencer une expérience, il faut savoir si les conclusions seront applicables à une (certaines) situation(s) spécifique(s) ou si les conclusions sont d'ordre général, c'est-à-dire qu'elles sont vraies quelle que soit la situation.

Il est d'usage de tirer une série de conclusions d'ordre spécifique avant de pouvoir tenter une généralisation. Dans le cas présent, nous avons mis l'accent sur 2 patrons de conception classiques. Sachant qu'il en existe plus d'une centaine, dont 23 classiques, le pouvoir de généralisation de l'expérience est donc limité.

De plus, une grande quantité de notions interviennent dans la compréhension d'un logiciel. Nous avons choisi de nous focaliser sur la compréhension via la consultation de diagrammes de classes UML, mais un programmeur tentant de comprendre un logiciel ne se limite évidemment pas à cette partie de la documentation.

L'expérience est donc catégorisée comme spécifique. Ce type d'expérience n'en est pas pour autant inutile car, comme nous l'avons dit plus haut, elles sont nécessaires à une généralisation future ainsi qu'à une meilleure compréhension du domaine.

Définition du contexte

La définition du contexte peut donc être résumée comme suit :
- Hors ligne
- Étudiant et professionnel
- Problème de taille réduite
- Spécifique

7.2.2 Formulation des hypothèses

Le but de l'expérience est de tester les hypothèses nulles suivantes et de tenter de les réfuter au profit des hypothèses alternatives présentées juste après :

- H_{0_1} : l'impact du patron de conception Observateur, sur l'effort moyen déployé pour l'exécution d'une tâche de maintenance est le même pour un groupe de sujets experts que pour un groupe de sujets novices.
- H_{0_2} : l'impact du patron de conception Objet composite, sur l'effort moyen déployé pour l'exécution d'une tâche de maintenance est le même pour un groupe de sujets experts que pour un groupe de sujets novices.

Si ces hypothèses nulles sont rejetées, nous pouvons supposer (sous réserve des menaces à la validité, chapitre 9) que certaines hypothèses alternatives suivantes sont vérifiées :

- $H_{\alpha_{1.1}}$: la présence du patron de coneption Observateur, dans le cadre d'une tâche de maintenance, diminue l'effort moyen d'un groupe de sujets experts par rapport à un groupe de sujets novices.
- $H_{\alpha_{1.2}}$: la présence du patron de conception Observateur, dans le cadre d'une tâche de maintenance, augmente l'effort moyen d'un groupe de sujets experts par rapport à un groupe de sujets novices.
- $H_{\alpha_{2.1}}$: la présence du patron de coneption Objet composite, le cadre d'une tâche de maintenance, diminue l'effort moyen d'un groupe de sujets experts par rapport à un groupe de sujets novices.
- $H_{\alpha_{2.2}}$: la présence du patron de conception Objet composite, dans le cadre d'une tâche de maintenance, augmente l'effort moyen d'un groupe de sujets experts par rapport à un groupe de sujets novices.

7.2.3 Sélection des variables

Les variables indépendantes sont les variables fixées ou contrôlées par l'expérimentateur. Les variables dépendantes sont les variables influencées par les variables indépendantes qui permettront

éventuellement de rejeter les hypothèses nulles.

Variables indépendantes

Nous identifions 2 variables indépendantes principales dans les hypothèses : elles représentent la présence ou l'absence d'un patron de conception et l'expertise du sujet :

- Présence d'un patron de conception : sa valeur représente le patron de conception présent dans le diagramme considéré. Les valeurs possibles pour cette variable sont « Observateur », « Composite » ou « Sans ». Le traitement sans patron de conception servira de référence pour voir l'évolution lorsque l'on introduit un patron.
- Expertise : nous classons les sujets en 2 groupes. Un groupe d'experts et un groupe de novices. Cette classification se fait sur base des parcours oculaires des sujets et des réponses fournies (voir section 8.3).

Nous retenons 2 variables confondantes à étudier afin de mieux comprendre les résultats de l'expérience :

- Sommeil : pour chaque sujet, les 2 informations suivantes ont été récoltées : le nombre d'heures de sommeil de la nuit passée et le nombre d'heures de sommeil habituellement nécessaires pour une nuit. La raison de ce choix est que la fatigue a un impact sur nos capacités cognitives et il est intéressant de voir si cet impact est significatif dans le cas de cette expérience.
- Statut : étant donné que des professionnels ont pris part à l'expérience, il semble intéressant de regarder s'il existe une différence notable entre ceux-ci et les étudiants. Nous refusons cependant d'associer la notion d'expert à celle de professionnel. Dans l'absolu, rien ne dit qu'un étudiant (notamment de 3^{e} cycle) n'a pas plus d'expérience avec les patrons de conception qu'un (jeune) professionnel. Les valeurs possibles pour cette valeur sont donc : Étudiant et Professionnel.

Variables dépendantes

Les variables dépendantes sont choisies en fonction des hypothèses et des données brutes fournies par un système d'oculométrie. Sont mesurés pour les analyses le taux de réponses correctes et différents types d'efforts (voir ci-dessous) fournis par le sujet.

La littérature donne principalement 2 types de mesures. Celles donnant une indication de l'effort de recherche de l'information et celles donnant une indication sur l'effort d'interprétation ou cognitif de l'information (section 6.2.4). En plus du pourcentage de réponses correctes, 2 mesures de chaque type ont été retenues.

Rappelons qu'une zone d'intérêt est une classe dans le diagramme de classes et qu'une zone d'intérêt pertinente est une classe contenant une partie de la réponse.

Comme indicateur de l'effort de recherche, nous utilisons :

- Densité spatiale (DS) : couverture d'un diagramme par la recherche du sujet. Elle peut être capturée par la distribution spatiale des fixations. Plus la distribution est uniforme, plus la recherche est extensive et inefficace, alors qu'une distribution ciblée reflète une recherche efficace. Elle permet de comparer des diagrammes entre eux. Une plus faible couverture indique une meilleure efficacité (organisation) du diagramme. Si l'on compare des sujets sur un même diagramme, c'est un indicateur de l'efficacité de la recherche d'un sujet par rapport à un autre. On s'attend à ce qu'un expert ait une couverture plus faible que celle d'un novice.
  Le diagramme est divisé en une grille. La densité spatiale est le nombre de cellules contenant au moins une fixation divisé par le nombre total de cellules de la grille. Pour notre expérience,

la taille de cellule utilisée est de 64 pixels.

$$DS = \frac{\sum_{i=1}^{n} c_i}{n} \tag{7.1}$$

$n =$ nombre de cases de la grille.
$c_i =$ 1 si la case numéro i est visitée, 0 sinon.

– La matrice de transition (MT) : exprime, pour un sujet, la fréquence des transitions d'une zone à une autre. Cette mesure considère les zones de recherche et les mouvements dans le temps. Cette mesure ajoute la composante temporelle de la recherche. La fréquence des transitions d'une région à une autre indique l'inefficacité de la recherche. Pour notre expérience, la taille de cellule utilisée est de 64 pixels.
Comme pour la densité spatiale, cette mesure donne une indication de l'efficacité d'un diagramme par rapport à un autre ou de l'efficacité d'un sujet par rapport à un autre. On s'attend à ce que la valeur de cette mesure pour un expert soit inférieure à celle d'un novice.

$$MT = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} c_{i,j}}{n \cdot n} \tag{7.2}$$

n = nombre de cases de la grille.
$c_{i,j} =$ 1 s'il existe une saccade allant de la case i à la case j, 0 sinon.

Comme indicateur de l'effort cognitif, nous utiliserons :

– Temps de fixation moyen (TFM) : plus une fixation est longue, plus le sujet passe du temps à interpréter l'objet fixé. De manière générale, les objets exigeant de longues fixations sont moins expressifs pour le sujet que ceux avec de courtes durées de fixation. Pour notre expérience, nous nous attendons à ce qu'un expert, lorsqu'il travaille sur un diagramme avec un patron de conception, focalise son attention sur les zones d'intérêt pertinentes et que celui-ci ait une charge cognitive plus faible et donc un temps de fixation moyen plus faible qu'un novice.

$$TFM = \frac{\sum_{i=1}^{f} tf_i}{f} \tag{7.3}$$

f = nombre de fixations
$tf_i =$ temps de la fixation i

– Rapport de fixations(on target all) (RFO) : nombre de fixations dans les zones d'intérêt divisé par le nombre total de fixations. C'est une mesure de la pertinence de l'effort d'un sujet et, par extension, de la charge cognitive du sujet. Un sujet avec un effort non pertinent devra fournir plus d'effort. Cet indicateur ne mesure pas l'effort absolu mais l'effort relatif propre au sujet. L'effort absolu doit plutôt être recherché dans le nombre de fixations ou le nombre de fixations dans les zones d'intérêt pertinentes.
Si l'on compare des diagrammes, un rapport élevé pour cette variable indique une efficacité du diagramme. Nous nous attendons à ce que les experts produisent, sur les diagrammes avec un patron de conception, un effort plus pertinent que celui des novices et aient donc une valeur plus élevée pour cet indicateur.

$$RFO = \frac{\sum_{i}^{f} fp_i}{f} \tag{7.4}$$

f = nombre total de fixations.
$fp_i =$ 1 si la fixation i est dans une zone pertinente, 0 sinon.

7.2.4 Sélection des sujets

L'expérience a été réalisée avec 21 sujets bénévoles :
- 8 sont des professionnels de la société Pyxis.
- 1 est un professionnel nous ayant accordé de son temps libre.
- 2 sont du 2e cycle de faculté d'informatique des FUNDP.
- 2 sont du 1er cycle de la section d'ingénierie logicielle de l'École Polytechnique de Montréal.
- 2 sont du 3e cycle de la section d'ingénierie logicielle de l'École Polytechnique de Montréal.
- 3 sont du 2e cycle du département d'informatique de l'Université de Montréal.
- 3 sont du 3e cycle du département d'informatique de l'Université de Montréal.

Tous les sujets sont familiarisés avec la conception « orienté-objet » ainsi qu'avec la notation UML. En ce qui concerne les patrons de conception, tous en ont des notions plus ou moins avancées suite à des cours ou des projets de développement.

7.2.5 Conception de l'expérience

La conception utilisée est très simple. Elle fait intervenir 1 facteur et 3 traitements. Le facteur est le patron de conception présent. Les 3 traitements sont : Sans patron, Observateur et Objet composite. Les 3 traitements sont appliqués à tous les sujets dans un ordre aléatoire pour éviter d'en favoriser un. La conception de l'expérience est illustrée par la figure 7.1.

Sujet	Sans patron	Observateur	Objet composite
Sujet 1	✓	✓	✓
Sujet 2	✓	✓	✓
Sujet 3	✓	✓	✓
...	...	...	...

TAB. 7.1: Conception des tests de l'expérience.

7.2.6 Instrumentation

Cette section décrit les différents objets nécessaires à notre expérience et donne quelques explications sur leur conception et les choix effectués.

Sélection des patrons de conception

Les critères de sélection des 2 patrons de conception pour notre expérience sont : leurs qualités, leur fréquence d'utilisation et leur complexité.

Qualités : suivant la statégie présentée dans la section 3.1.3, nous nous basons sur le sondage [KG08] (voir section 5.3) concernant les qualités des patrons de conception classiques. Il semblerait que 7 patrons cumulent 7 des 10 qualités repertoriées par les auteurs du sondage. Ces 7 patrons sont donnés dans le tableau 7.2.

Complexité : il faut que les patrons sélectionnés ne soient pas trop simples. Typiquement, un patron pour lequel le comportement est encapsulé dans une classe n'est pas fort mis en avant avec un diagramme de classe et il peut devenir difficile de détecter l'influence de ce type de patrons avec un oculomètre.

Fréquence d'utilisation : afin d'augmenter nos chances de trouver des sujets « experts » avec les patrons sélectionnés, il vaut mieux choisir ces derniers parmi les patrons les plus fréquemment utilisés. Le site de la « Data & Object Factory [2] » donne une idée de la fréquence d'utilisation

[2] `http://www.dofactory.com/Patterns/Patterns.aspx`

des 23 patrons de conception classiques (voir tableau 7.2). Afin de connaître la source de cette information, nous avons tenté de contacter les auteurs du site, mais sans succès.

Patron de conception	Fréquence
Itérateur	5/5
Observateur	5/5
Objet composite	4/5
Stratégie	4/5
État	3/5
Chaîne de responsabilité	2/5
Interpréteur	1/5

TAB. 7.2: Fréquence d'utilisation des 7 patrons de conception envisagés.

Parmis les 7 patrons envisagés, le patron Itérateur est considéré comme trop simple, car composé d'une seule classe ; il n'a donc pas été utilisé. Les patrons de conception État et Stratégie ont le même diagramme de classes et nous les considérons de ce fait comme des cas particuliers qu'il est préférable d'étudier une fois le domaine mieux connu. Ils ne sont dès lors pas utilisés. Suivant les fréquences d'utilisation renseignées, notre choix se porte sur l'Observateur et sur l'Objet composite [3].

Définition des *stimuli*

Les diagrammes (voir annexe C) sont issus de logiciels réels. Un expérimentateur créant de toute pièce ses diagrammes s'expose trop fortement [4] au risque d'influencer le résultat de l'expérience vers les résultats qu'il espère. Les diagrammes sont recréés à l'aide d'un outil d'ingénierie inverse. Les classes composant les patrons de conception sont alors replacées selon la disposition canonique présentée dans [GHJV95] de façon à favoriser le processus de la mémoire de travail (voir section 2.3.2). Le reste des classes est alors disposé de manière à être présentable et surtout lisible sur un écran d'ordinateur. Cette contrainte ne nous a pas permis d'utiliser les outils de disposition automatique des classes.

Avant de présenter le diagramme au sujet, nous lui proposons une petite mise en contexte. Cette mise en contexte a pour but de fournir quelques informations au sujet, de manière à ce qu'il ne soit pas surpris par le diagramme qui lui est présenté. Chaque mise en contexte propose 3 types d'information : le nom du système d'où nous avons tiré le diagramme, une brève présentation des fonctionnalités du système, enfin une petite présentation de la partie du sytème qui nous intéresse. Ces mises en contexte sont disponibles dans l'annexe C.

Afin de laisser l'opportunité au sujet de relire la question, nous la plaçons au-dessus du diagramme car l'œil a tendance à regarder le coin supérieur gauche en premier lieu [YKM07]. Afin de faire ressortir un peu plus la question, nous l'affichons en rouge. Nous présentons tous les *stimuli* en anglais pour permettre la participation de sujets non francophones. Les *stimuli* devant être identiques pour chacun, il est exclu d'en avoir en français et d'autres en anglais. Par contre, le sujet a le choix entre le français et l'anglais pour fournir sa réponse. Ceci afin de modifier le moins possible le schéma cognitif suivi par le sujet devant traduire sa réponse avant de la donner.

Précisons qu'il n'est pas évident de trouver la bonne granularité pour les questions. Une question trop précise donne trop d'éléments de réponse tandis qu'une question trop floue peut être interprétée de plusieurs manières. De plus, la complexité des 3 tâches doit être plus ou moins équivalente

[3] Même l'information concernant la fréquence n'a pu être confirmée, elle ne nous semble pas dénuée de sens : les interfaces graphiques utilisent beaucoup le patron Observateur et beaucoup de structures de données utilisent l'Objet composite.

[4] Même si c'est inconscient.

afin de permettre la comparaison. Malheureusement, la complexité d'une tâche est extrêmement difficile à mesurer Les questions ont donc été étudiées à l'aide du professeur Yann-Gaël Guéhéneuc.

La représentation choisie pour les diagrammes est UML (voir section 6.1). Plusieurs autres représentations ont été proposées, notamment [DYZ07] qui, selon [Por08], est plus efficace en présence de patrons de conception. Mais cette représentation a le désavantage d'avantager les connaisseurs des patrons de conception et nécessite de surcroît un apprentissage supplémentaire. Enfin, comme l'expérience utilise aussi un diagramme sans patron de conception, cette représentation n'est dès lors plus pertinente.

Sélection des systèmes

Afin de minimiser le biais dû à la connaissance des systèmes ainsi que le biais d'apprentissage (voir chapitre 9), chaque *stimulus* est conçu à partir d'un système différent. Nous utilisons des systèmes libres afin d'avoir accès aux sources pour effectuer une ingénierie inverse. Les logiciels présents dans la base de données de l'équipe Ptidej [5], pour lesquels une détection des patrons de conception a été effectuée et qui implémentent au moins un des 2 patrons étudiés, sont au nombre de 7 :

- JHotDraw
- JUnit
- Lexi alpha
- MapperXML
- Netbeans
- PMD
- QuickUML

Parmi ces logiciels, JHotDraw est écarté car déjà utilisé par [Jea08], ce qui risque d'introduire un biais d'apprentissage, ainsi que Netbeans car trop gros pour effectuer une ingénierie inverse. Le reste de la sélection est assez subjective. Finalement, JUnit [6] est utilisé pour le diagramme comprenant le patron Objet composite, QuickUML [7] pour le diagramme comprenant le patron Observateur et un troisième logiciel, ArgoUML [8], pour le diagramme ne comprenant pas de patron de conception.

Checklist de réponses

Ne pouvant pas bouger, les sujets doivent donner une réponse orale. Afin de prendre note, nous avons défini une réponse correcte pour chaque question. Il nous suffit alors de cocher les éléments de réponse donnés par le sujet. Malheureusement, cette approche n'est pas aussi concluante que prévu, dans la mesure où il n'existe pas qu'*une* réponse correcte aux questions. En effet, beaucoup de réponses reçues sont des réponses correctes, ou au moins qui « fonctionnent », alors que le sujet n'utilise pas à bon escient les éléments déjà en place (*e.g.*, au lieu travailler avec une classe abstraite existante, le sujet travaille avec chacune des sous-classes existantes). Du point de vue de la maintenance, cependant, ces réponses posent un problème à plus long terme car elles provoqueraient beaucoup de changements. Malgré cela, au vu de la question posée, elles ne peuvent pas être considérées comme fausses. Pour notre analyse, nous considérons dès lors 2 types de réponses correctes : les réponses correctes respectant la sémantique du diagramme et les réponses correctes ne respectant pas la sémantique du diagramme.

[5] `http://www.ptidej.net/`
[6] `http://www.junit.org/`
[7] `http://sourceforge.net/projects/quj/`
[8] `http://argouml.tigris.org/`

Définition de la notion d'expert

Nous voyons 3 possibilités pour évaluer l'expertise d'un sujet :

1. Le nombre de projets ou les années de programmation, etc.
2. L'auto-évaluation sur une échelle déterminée.
3. Le résultat à un test.

L'expérience : on évalue l'expertise du sujet par rapport à son utilisation antérieure des patrons de conception. Cela peut se faire selon le nombre d'années d'utilisation, le nombre de projets, etc. Le problème avec cette évaluation, est qu'elle ne prend pas correctement en compte les connaissances réelles du sujet. Chaque projet étant différent, l'utilisation qu'ils font des patrons de conception l'est aussi. De plus, certains sujets particulièrement doués peuvent obtenir le même niveau de connaissance beaucoup plus rapidement que d'autres.

L'auto-évaluation : le sujet se note lui-même sur une échelle pré-définie. Ce type d'évaluation nous semble sujet à un biais introduit par le sujet. Un sujet particulièrement confiant en lui pourrait avoir tendance à se surévaluer tandis qu'un sujet particulièrement peu confiant en lui pourrait avoir tendance à se sous-évaluer. De plus, comme nous utilisons des patrons de conception fréquents, il n'est pas impossible que l'expérience d'un sujet fasse qu'il est familier avec un patron de conception mais sans le savoir.

Résultat à un test : l'expérimentateur prépare un test qu'il fait passer à chaque sujet. La note attribuée au test est alors l'évaluation de l'expertise du sujet. Bien que cette solution semble plus adaptée que les 2 précédentes, elle est en pratique compliquée à mettre en place. Il faut s'assurer que les questions du test ne soient pas divulguées par les premiers sujets. Réunir l'ensemble des sujets afin de passer le test en même temps, n'est pas envisageable vu la difficulté avec laquelle les sujets sont recrutés. Il n'est pas très diplomate non plus de faire passer un test supplémentaire à des professionnels. Ajoutons que la question compliquée de l'objectivité du test doit être posée. Enfin, une telle démarche donnerait des éléments permettant au sujet de tenter de deviner le but de l'expérience et donc d'altérer son comportement.

Aucune de ces possibilités ne convient pour la création des groupes. Plutôt que de créer un groupe d'experts et un groupe de novices et d'ensuite voir s'ils réagissent différemment, nous avons décidé de prendre le problème dans l'autre sens. Nous avons décidé de d'abord créer 2 groupes réagissant différemment à l'aide d'une technique de classification et d'ensuite voir si un groupe correspond plus à un groupe d'experts et l'autre à un groupe de novices. De cette manière, les groupes ne sont formés qu'à partir de données récoltées et non plus à partir de considérations susceptibles d'introduire un biais. Cette classification est décrite dans l'analyse des données (section 8.3).

Recrutement

Le principe le plus simple pour récolter les inscriptions est de mettre des affiches, d'envoyer des courriels et de faire un site internet d'inscription. C'est donc la manière retenue. Suite à l'inscription sur le site, le sujet reçoit un courriel de confirmation ainsi que quelques conseils afin de passer l'expérience dans les meilleures conditions (*e.g.*, ne pas consommer d'alcool la veille, dormir un nombre d'heures suffisant, ne pas mettre de maquillage au niveau des yeux). L'annexe E propose l'affiche d'inscription et la page d'accueil du site internet.

Le formulaire

Afin de récolter les informations concernant l'état de fatigue du sujet, nous remplissons un formulaire. Cette information est nécessaire à l'expérience pour 3 raisons :

- La première a déjà été évoquée dans la section 7.2.2. Il s'agit d'un facteur que nous pensons confondant et que nous voulons étudier.
- La seconde est que la fatigue influence l'effort de calibration nécessaire. Un sujet fatigué aura le regard fuyant et l'expérimentateur aura plus de difficultés à calibrer le matériel.
- La dernière raison est qu'une trop grosse fatigue est, selon nous susceptible d'expliquer certaines données aberrantes détectées lors de l'analyse.

Selon [CDEDSAT05], il existe plusieurs symptômes de la fatigue chez une personne :

- lassitude,
- somnolence, notamment s'endormir contre sa volonté (« micro-siestes »),
- irritabilité,
- dépression,
- étourdissement,
- perte d'appétit,
- troubles digestifs
- prédisposition accrue aux maladies.

Malheureusement, demander des informations relatives à ces symptômes est trop intrusif, ce que nous ne voulons pas. À la place, nous récoltons simplement 4 informations : le nombre d'heures de sommeil d'une nuit normale, le nombre d'heures de sommeil de la nuit passée, l'heure de réveil et l'heure de passage du sujet.

Tutoriel

Avant de faire passer l'expérience proprement dite aux sujets, nous leur proposons un tutoriel. Ce tutoriel comprend plusieurs informations. Premièrement, le but de l'expérience est exposé. Nous nous contentons de dire que l'expérience vise à mieux comprendre le comportement d'un programmeur lorsqu'il examine un diagramme de classes UML. Nous ne mentionnons pas les patrons de conception pour ne pas influencer leur comportement et ne pas introduire de biais (voir chapitre 9). Suivent l'explication du fonctionnement global de l'oculomètre, un rappel concernant les diagrammes de classes UML, enfin une mise en situation durant laquelle le sujet est invité à se comporter comme il le fera lors de l'expérience – à la différence près qu'il peut poser les questions qu'il se pose. Bien entendu, nous donnons quelques recommandations de base, comme ne pas bouger la tête.

Le but de ce tutoriel est triple : limiter l'appréhension du sujet et son stress, donner toutes les informations nécessaires avant le début de l'expérience afin de ne pas avoir à intervenir durant le déroulement de celle-ci et, enfin, évaluer si le sujet a bien les connaissances de base concernant les diagrammes de classes UML.

Le premier but est de limiter au maximum l'appréhension du sujet concernant l'expérience. Un sujet sous l'influence du stress est un sujet qui ne réagit par normalement. Il existe plusieurs manières de réagir lorsque l'on est stressé. Certains peuvent simplement avoir envie d'en finir au plus vite et non plus de répondre correctement aux questions. D'autres voudront surtout ne pas faire d'erreurs et resteront très (trop) longtemps afin de s'assurer de leurs réponses, etc. Dans tous les cas de figure, il ne s'agit pas là du comportement normal du sujet. Or, c'est le comportement normal que nous voulons étudier.

Le deuxième but est de donner au sujet toutes les informations dont il a besoin avant que l'expérience ne commence réellement. Afin de ne pas influencer différemment les sujets, les informations données pendant le déroulement de l'expérience doivent être exactement les mêmes pour tous. Pour y arriver, nous choisissons la manière la plus simple : n'en donner aucune. C'est pourquoi, il faut faire attention à ce qu'ils reçoivent préalablement toutes les informations utiles.

Enfin, le seul prérequis pour la participation à notre expérience est d'avoir une certaine connaissance de base concernant les diagrammes de classes UML. Les ressources manquant pour vérifier

finement ces connaissances, ce tutoriel permet de les vérifier grossièrement. Nous avons refusé un sujet suite à cette vérification.

Procédure d'exécution

Comme expliqué plus haut, les informations données aux sujets doivent être identiques. Il en va de même pour le traitement (en terme d'accueil, etc.) du sujet. C'est pourquoi une procédure pour l'accueil des sujets ainsi que pour l'expérimentation proprement dite est respectée. Idéalement, ce n'est pas l'expérimentateur mais une autre personne qui doit se charger de l'accueil des sujets et de l'exécution de l'expérience : nous avons alors un collecteur de données complètement indépendant, ne pouvant pas, étant donné sa « non-compétence », influencer les sujets. Malheureusement, l'effectif réduit ne le permet pas. Nous minimisons au maximum les différences entre les sujets lors de cette étape par le respect d'une procédure représentée sous forme d'un organigramme décrivant la suite des actions à effectuer et des sujets à aborder. Nous affichons cet organigramme au format $A0$ dans la salle accueillant l'expérience. Cette procédure a été mise en place par nos prédécesseurs et une reconstitution de l'organigramme est disponible dans l'annexe F. La figure 7.1 en donne une simplification.

FIG. 7.1: Enchaînement des étapes lors de l'accueil d'un sujet.

L'accueil consiste principalement à vérifier l'identité du sujet et à s'assurer que le sujet est apte à prendre part à l'expérience.

Le tutoriel est expliqué ci-dessus.

L'expérience consiste à présenter les *stimuli* aux sujets et à récolter les données (voir section 7.3).

Le formulaire consiste à poser des questions concernant la fatigue et l'expérience du sujet.

Les remerciements consistent en des remerciements et le don d'un petit chocolat belge.

Oculomètre

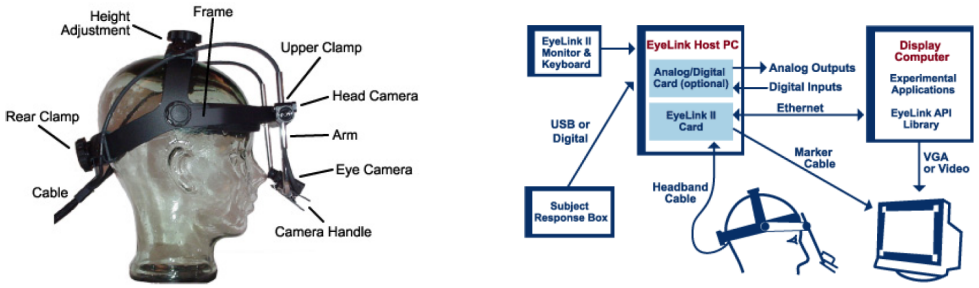
Eye Link II est l'oculomètre utilisé pour notre expérience. Il est proposé par la société SR Research [9] qui se base sur la vidéo pour enregistrer les mouvements oculaires du sujet lorsqu'il examine un diagramme de classes UML afin de comprendre le système modélisé. Cela permet d'étudier de manière relativement non intrusive et complètement inoffensive le comportement extérieur d'un programmeur impliqué dans une activité de compréhension logicielle.

Grâce à un casque, cet oculomètre enregistre, sans interférer avec l'activité du sujet, les données relatives aux coordonnées des mouvements oculaires de celui-ci sur un écran d'ordinateur. Le système convertit ensuite les coordonnées brutes en fixations et saccades. Le système Eye Link II est composé de 2 ordinateurs et d'un casque.

La figure 7.2a montre le casque et en donne les différentes parties. Une caméra permet de positionner la tête par rapport à l'écran pendant que d'autres caméras enregistrent les mouvements oculaires. Les autres composants du casque sont utilisés pour faire tenir les caméras et le casque.

La figure 7.2b illustre l'acquisition des données au travers des 2 ordinateurs et du casque. L'ordinateur d'affichage est l'ordinateur qui affiche les *stimuli* au sujet. Il est relié à l'ordinateur hôte par un cable en fibre optique afin de permettre la synchronisation avec le *stimulus* affiché. L'ordinateur hôte est la « console » du chercheur. Il est relié à l'ordinateur d'affichage par un cable en fibre

[9] http ://sr-research.com/



(a) EyeLink II : Illustration des parties du casque.

(b) EyeLink II : Configuration du système.

FIG. 7.2: EyeLink II (Tiré de [SR 06]).

optique. Cette liaison couplée à une API permet à l'ordinateur hôte de calibrer les caméras à l'aide de 4 capteurs disposés aux 4 coins de l'écran d'affichage, d'afficher l'écran de l'ordinateur d'affichage au chercheur et de générer les données. Les données sont enregistrées sur l'ordinateur hôte et, à la fin de l'expérience, transférées sur l'ordinateur d'affichage pour les analyses ultérieures.

EyeLink II exécute un petit programme écrit en C pour chaque expérience. Ce programme organise les différents événements pendant l'expérience. L'exécution de l'expérience est donc exactement la même pour chaque sujet, à l'exception de l'ordre de présentation des *stimuli* qui est aléatoire. La figure 7.3 illustre l'exécution de notre programme. L'action « Pression escape » permet au sujet de signaler qu'il est prêt à continuer.

Pour finir, précisons que l'expérience est prévue pour être la plus courte possible. Le poids du casque devenant gênant après un certain temps.

Taupe

Taupe « Thoroughly Analysing the Understanding of Programs through Eyesight » est un logiciel initialement développé par Yann-Gaël GUÉHÉNEUC [Gué06] permettant l'analyse des données fournies par l'oculomètre « Eye link II ». Nous en avons ré-écrit une partie car la succession d'étudiants a rendu le code compliqué et incohérent (*e.g.*, redondances). De plus, nous voulions lui ajouter quelques fonctionnalités. L'annexe B en propose une brève description.

7.2.7 Validité de l'expérience

Une analyse de la validité de l'expérience a bien entendu été réalisée durant la phase de planification mais cette analyse a été complétée durant tout le processus de l'expérience jusqu'à l'analyse des données. C'est pourquoi cette analyse est présentée dans le chapitre 9.

7.3 Exécution de l'expérience et validation des données

Cette section explique la manière dont les données sont récoltées et validées. Nous y expliquons l'accueil des sujets, le passage de l'expérience par le sujet ainsi que le processus de validation des données.

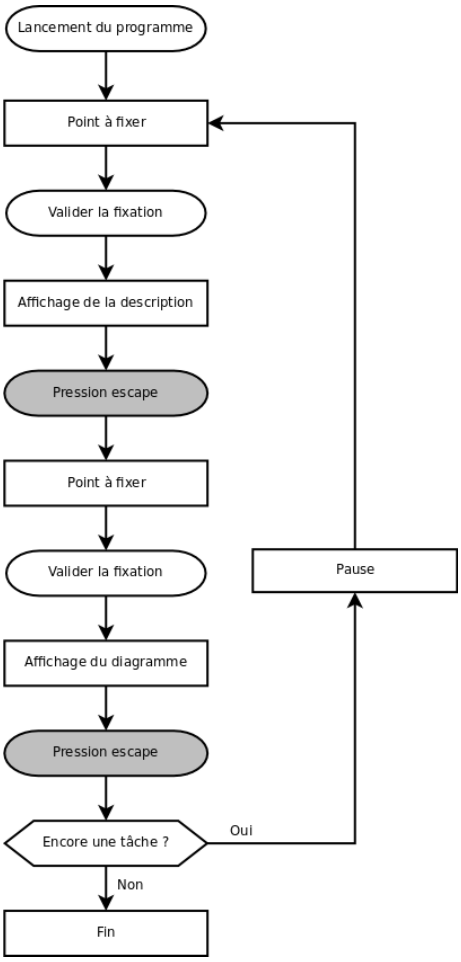


FIG. 7.3: Ordinogramme présentant l'exécution des tests de l'expérience. Les rectangles sont des événements ne nécessitant pas d'action humaine. Les bulles grises sont les événements nécessitant une action du sujet, les autres bulles sont les événements nécessitant une action de l'expérimentateur.

7.3.1 Exécution de l'expérience

L'exécution de l'expérience suit le canevas décrit dans la section 7.2.6. Nous détaillons dans cette section l'accueil des sujets, la calibration de l'oculomètre et le déroulement des tests.

Accueil

Nous avons expliqué dans la section précédente les mesures prises pour tenter de mettre les sujets dans les meilleures conditions. Toutes ces mesures sont inefficaces si l'attitude de l'expérimentateur est de nature à mettre les sujets mal à l'aise. Nous prenons donc soin d'être bien à l'écoute des interrogations des sujets. Nous leur précisons également qu'ils peuvent abandonner l'expérience à tout moment sans conséquence, que le système est inoffensif et que les données sont rendues anonymes.

L'expérience a eu lieu dans 2 endroits différents. Nous avons rencontré les étudiants à l'École Polytechnique et nous avons déplacé le matériel dans les locaux de l'entreprise Pyxis afin que certains de ses employés puissent prendre part à l'expérience.

Afin de laisser le moins possible de facteurs extérieurs influencer l'expérience, nous avons utilisé à l'École Polytechnique une pièce calme, fermée, arborant une affiche demandant de ne pas déranger. Cette pièce calme est une condition *sine-qua-non* car chaque bruit dérangeant le sujet peut lui faire bouger les yeux et avoir une implication directe sur les données récoltées. Lorsque nous étions dans les locaux de Pyxis, nous disposions également d'une pièce calme. Ayant pris soin de faire savoir qu'une expérience s'y déroulait, nous n'avons naturellement pas été dérangés.

Calibration

La calibration est la première partie du passage effectif de l'expérience. Elle permet de mettre en place les paramètres nécessaires à l'appareil pour calculer la position des yeux sur l'écran. Pour le sujet, elle consiste à fixer une dizaine de points. L'oculomètre enregistre la position des yeux et détermine les paramètres qui lui permettront de déduire les mouvements du regard du sujet. Une autre dizaine de points à fixer permettent de tester le résultat de la calibration et déterminer si elle s'est bien déroulée. L'écran de l'expérimentateur affiche cette seconde dizaine de points à fixer et affiche également la position calculée du regard du sujet. Il n'est pas difficile de se rendre rapidement compte de la qualité de la calibration d'un sujet. Ensuite, l'appareil sélectionne l'œil le plus précis du sujet. Ce sont les données générées par cet œil qui seront enregistrées. Il n'est pas rare de devoir effectuer cette opération plusieurs fois avant d'obtenir une calibration correcte. Il nous est arrivé de devoir refuser un sujet car il nous était impossible de calibrer l'appareil pour celui-ci. Cette impossibilité peut se produire lorsque le contour des yeux est fort foncé, ce qui était le cas. L'appareil tente de détecter le noir de la pupille de l'œil, qu'il confond alors avec le contour des yeux.

Tests

Les tests sont la deuxième partie du passage effectif de l'expérience. Le programme les organisant est décrit dans la section 7.2.6.

Si on examine le déroulement de l'expérience (figure 7.3), on constate que le sujet doit effectuer l'action « Pression escape ». Cela lui permet de piloter l'expérience. C'est lui qui décide à quel moment il estime avoir fini de lire la description proposée, comme c'est lui qui décide à quel moment il estime avoir fini de répondre à la question concernant le diagramme affiché. La fixation se trouvant entre chaque affichage (description et diagramme) permet au matériel d'enregistrer une petite correction dans le cas où le sujet a bougé la tête. Entre chaque tâche, l'expérimentateur propose au sujet de faire une courte pause. Cette pause permet au sujet de fermer et donc de reposer ses yeux quelques secondes. Malheureusement, il s'agit là de la seule chose autorisée au sujet. Il ne peut surtout pas bouger la tête, auquel cas, il faut repasser par l'ensemble du processus de calibration et recommencer l'expérience.

7.3.2 Validation des données

La validation des données doit se faire sous 2 angles. Premièrement, il faut vérifier qu'aucun sujet n'a tenté d'influencer les données, volontairement ou pas. Pour limiter ce risque, nous n'avons pas expliqué le but exact de l'expérience et nous sommes limités aux informations indispensables. Le sérieux de chacun des sujets et les informations limitées sur le but réel de l'expérimentation nous laissent penser qu'aucun sujet n'a tenté d'influencer volontairement les résultats de l'expérience.

Nous avons par contre rencontré un problème en ce qui concerne la validation technique des données. Le problème est dû aux mouvements de la tête des sujets. Lorsqu'un sujet bouge la tête, l'emplacement du regard enregistré est décalé par rapport à son emplacement réel. Ce phénomène

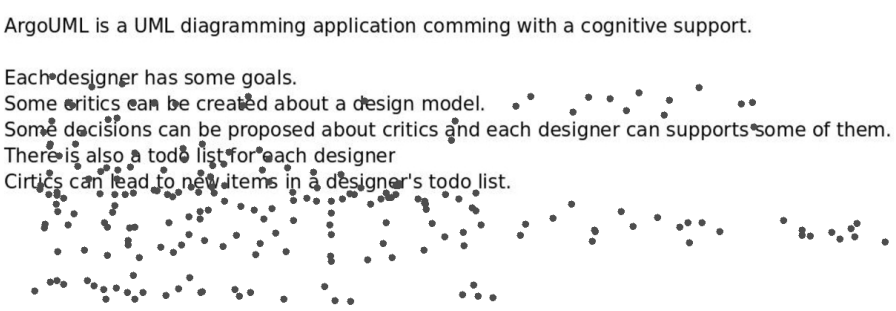


FIG. 7.4: Illustration du décalage présent dans les données enregistrées. Les points représentent les fixations enregistrées par l'oculomètre.

est fort bien illustré par la figure 7.4. Il s'agit de la représentation des fixations d'un sujet lors de l'examen de la description du diagramme sur ArgoUML.

Sur la figure 7.4 le décalage semble constant. Ceci est dû au fait que le temps d'examen de la description est court. L'examen d'un diagramme est plus long et les sujets effectuent plusieurs mouvements, ce qui fait que le décalage est permanent mais non sa valeur. Les données sont relativement imprécises, particulièrement lorsque le sujet en est au 3^{e} *stimulus* et qu'il est fatigué.

Avant d'être validées, les données doivent donc être corrigées pour limiter le problème. Elles peuvent être corrigées de 2 manières. La première consiste à visionner l'ensemble des tests, de tenter de détecter chaque mouvement de tête et d'introduire manuellement un décalage qui corrige le mouvement. Cette méthode est, selon nous, mauvaise. Détecter chaque mouvement de la tête n'est pas évident/possible (*e.g.*, une suite de petits mouvements indétectables mais qui au final ont un effet significatif). De plus, les données dans un diagramme de classes UML sont relativement denses. Il est difficile de déterminer, suite à un mouvement plaçant une fixation juste entre 2 zones d'intérêt, l'objet de l'attention. Une suite de mauvaises appréciations peut changer radicalement le résultat. Enfin, ce type de correction peut mener l'expérimentateur à arranger, consciemment ou pas, les données afin qu'elles collent à ce qu'il attend.

La seconde méthode, que nous avons utilisée, nous semble plus fiable. Elle consiste à introduire un décalage global pour chaque test. Pour ce faire, nous posons 3 hypothèses :

1. Chaque test doit contenir des fixations sur la question.
2. Il est possible qu'un test contienne des fixations sur des parties inintéressantes (c'est-à-dire hors des classes et liens), mais cela ne concerne pas la majorité des fixations.
3. Il est moins important de prendre en considération les fixations sur les relations entre les classes car, selon les conclusions de [Gué06], les sujets n'ont que fort peu tendance à s'attarder sur ce type d'information.

La correction des données consiste à ajouter, pour chaque test, un décalage global sur l'axe des x ainsi que sur l'axe des y. Suite aux 2 premières hypothèses, nous avons installé dans Taupe un compteur de fixations pour chaque classe et pour la question. Nous tentons ensuite de faire correspondre au maximum les fixations avec ces zones. Suite à la troisième hypothèse, nous ne prenons pas en compte les fixations sur les relations entre les classes. La disparité des fixations dûe aux mouvements de la tête rend de toute manière bien souvent cette tentative vaine. Il en résulte finalement que la disparité et la densité des fixations font qu'il n'est pas toujours évident de corriger les données correctement. La figure 7.4 illustre cette disparité et densité des fixations.

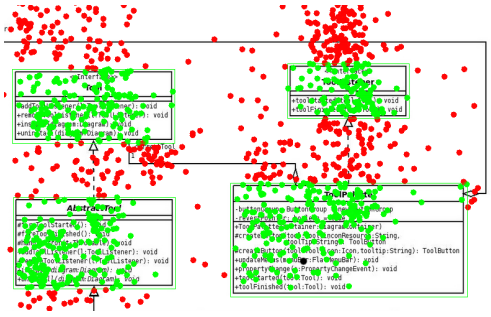


FIG. 7.5: Illustration de la difficulté de la correction des données. Les points représentent les fixations du sujet.

Chapitre 8

Analyse des données

Ce chapitre présente l'analyse des données. Nous commençons par une présentation générale des données dans la section 8.1 . Nous expliquons ensuite dans la section 8.2 la réduction de l'ensemble des données effectuée. La section 8.3 explique le procédé utilisé pour classer les sujets en 2 groupes. Les hypothèses principales sont analysées dans la section 8.4 et les facteurs confondants dans la section 8.5. Enfin, nous concluons l'analyse dans la section 8.6.

8.1 Généralités

Avant d'entrer dans le détail, le tableau 8.1 donne quelques statistiques sur les données collectées.

Données collectées	
Nombre de sujets	21
Taille des données (AD)	88 Mb
Nombre de vidéos	63
Temps total d'enregistrement (AD)	425 minutes
Nombre total de fixations	41896
Nombre total de fixations dans une AOI	28977
Nombre total de fixations dans une AOI pertinente	18438
Temps moyen pour Uunit	286 secondes
Temps moyen pour ArgoUML	229 secondes
Temps moyen pour QuickUML	270 secondes
Nombre moyen de fixations pour Junit (Objet composite)	714
Nombre moyen de fixations pour QuickUML (Observateur)	700
Nombre moyen de fixations pour ArgoUML (Sans patron)	579

Tab. 8.1: Statistique données collectées : aucune statistique ne prend en compte les descriptions présentes avant chaque tâche, hormis les 2 affichant la mention AD (avec description).

Avant d'aller plus loin dans l'analyse, voici d'abord un rappel des différentes mesures :

RFO : le rapport de fixation « on target : all » est une mesure de la pertinence de l'effort cognitif. Plus la valeur de cette variable est élevée, plus le sujet est efficace.

DS : la densité spatiale est une mesure de l'effort de recherche. Plus la valeur de cette variable est élevée, plus l'effort de recherche du sujet est élevé.

MT : la matrice de transition est une mesure de l'effort de recherche. Plus la valeur de cette variable est élevée, plus l'effort de recherche du sujet est élevé.

TFM : le temps de fixation moyen est une mesure de l'effort cognitif. Plus le sujet passe de temps sur une même information, plus l'effort cognitif du sujet est grand.

Rappelons que nous séparons les réponses correctes en 2 groupes (section 7.2.6) : les réponses respectant la sémantique du diagramme et les réponses ne la respectant pas.

En analysant la figure 8.1 qui présente la distribution des données pour les différentes mesures, il est possible de faire une première observation concernant les 2 types de réponses (voir figure 8.1e). Le taux de réponses sémantiquement incorrectes est très élevé pour la tâche sans patron de conception : il est de 75%. Par contre, le taux de réponses sémantiquement correctes pour cette tâche est très bas et est de 10%. La différence est de 60% alors qu'elle n'est que de 20% dans les 2 cas avec un patron de conception. Cette énorme différence (le triple) tend à montrer que même si les sujets donnent plus souvent une réponse correcte à la tâche sans patron, cette réponse n'est que rarement pleinement satisfaisante. Autrement dit, *les patrons de conception favoriseraient les réponses sémantiquement correctes.*

Pour les calculs des moyennes dans la suite de l'analyse, nous attribuons aux différentes réponses une valeur de :
- 1 pour une réponse correcte respectant la sémantique du diagramme.
- 0.5 pour une réponse correcte mais ne respectant pas la sémantique du diagramme.
- 0 pour un réponse fausse.

La figure 8.1f donne le pourcentage de réponses correctes avec ces valeurs.

La section 4.2.2 fournit les bases nécessaires à l'interprétation des graphiques et résultats des tests d'hypothèse présentés plus loin dans cette section. L'ensemble des résultats sont présentés ; néanmoins, dans un souci de concision, seuls les résultats intéressants sont développés. Pour obtenir les valeurs précises des données présentées par les graphiques, le lecteur peut se référer à l'annexe A.4.

8.2 Réduction de l'ensemble de données

Comme le montrent les figures 8.1a, 8.1b, 8.1c et 8.1d, le sujet numéro 6 se révèle avoir, dans 10 cas sur 12, des valeurs extrêmes. Cela représente plus de 80% des variables dépendantes (hors réponse) utilisées.

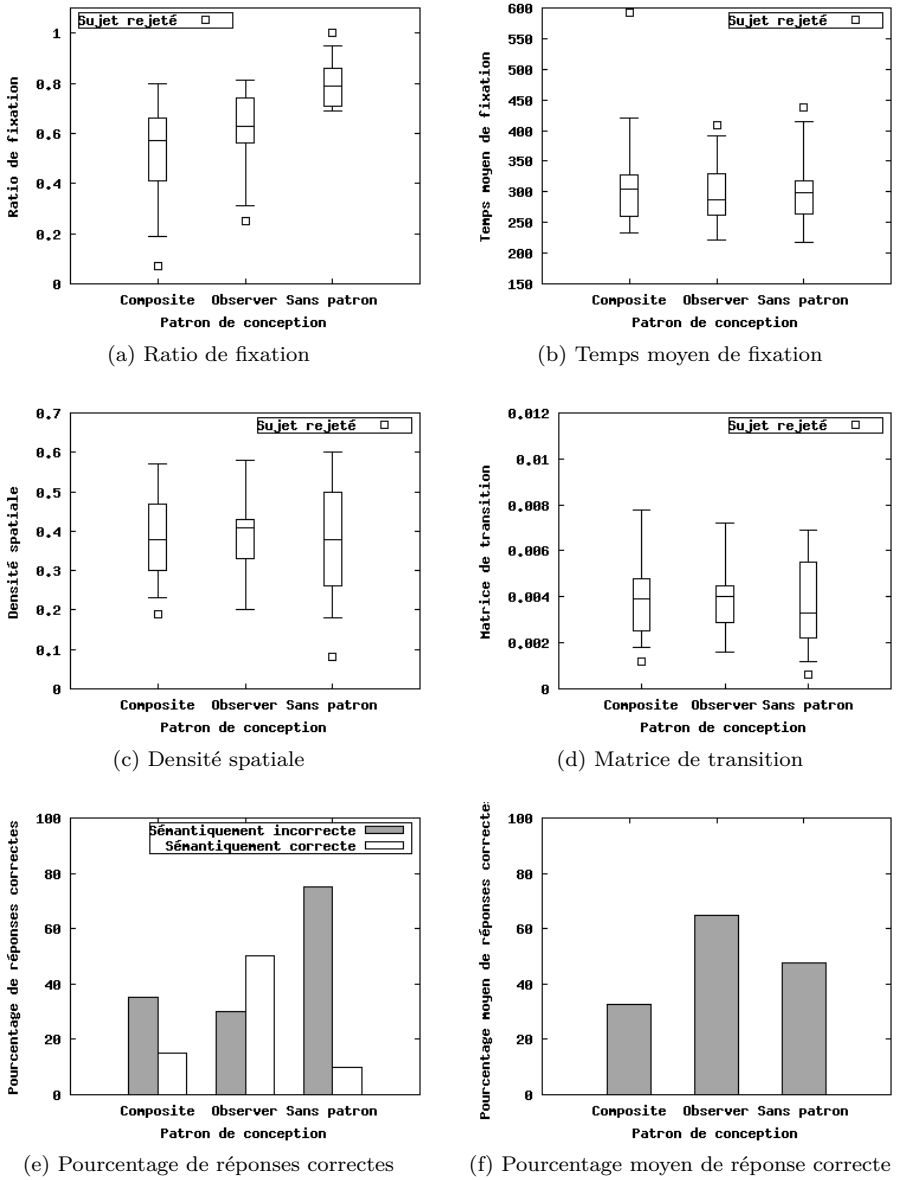
Sachant que les taux pour la mesure RFO sont extrêmes pour les 3 diagrammes et que le nombre de fixations du sujet est, pour 2 diagrammes, la valeur minimale et, pour le troisième, inférieur au premier quartile, nous pouvons nous demander si le sujet n'a pas simplement voulu répondre aux questions le plus vite possible. Les mesures DS et MT vont également dans ce sens. Des valeurs anormalement basses montrent bien que le sujet n'a pas effectué énormément d'efforts de recherche. Enfin, le pourcentage de bonnes réponses du sujet est inférieur à la moyenne, ce qui signifie aussi que nous n'avons pas affaire à un sujet surdoué n'ayant pas besoin d'énormément de recherche afin de trouver la bonne réponse. Les réponses de ce sujet au formulaire indique qu'il a dormi 1/2 heure de plus que d'habitude. La cause de ces valeurs extrêmes ne peut pas être expliquée par le manque de sommeil.

L'ensemble de ces considérations a fait que ce sujet est écarté de notre analyse [1]. L'ensemble des résultats fournis en annexe A ne tiennent donc pas compte de ce sujet.

8.3 Classification des sujets

Comme expliqué dans la section 7.2.6, nous voulons classer les sujets à partir de leur parcours oculaires. Pour classer les sujets en 2 groupes, nous avons décidé d'utiliser la technique de classification

[1] Les figure 8.1e et 8.1f sont créées sans avoir pris en considération ce sujet, contrairement aux figures 8.1a, 8.1b, 8.1c et 8.1d



(a) Ratio de fixation

(b) Temps moyen de fixation

(c) Densité spatiale

(d) Matrice de transition

(e) Pourcentage de réponses correctes

(f) Pourcentage moyen de réponse correcte

FIG. 8.1: Distribution des données pour les différentes mesures. Les figures a, b, c et d affichent également les données concernant le sujet écarté.

automatique AGNES (section 4.3.2) qui demande en entrée une simple matrice de dissimilarité. Nous voulons donc obtenir cette matrice à partir des parcours oculaires de nos 20 sujets. Nous devons donc obtenir un moyen de mesurer une dissimilarité entre 2 parcours oculaires réalisé sur un même *stimulus*. Pour y arriver, nous utilisons le plus petit polygone convexe qui entoure les zones du diagrammes ayant fait l'objet d'un certain seuil d'attention de la part du sujet. Le seuil d'attention est un pourcentage du nombre de fixations total d'un parcours oculaire.

Afin d'obtenir les zones d'un diagramme ayant fait l'objet d'un certain seuil d'attention (que nous appelons : zones d'attention), nous décomposons le diagramme en cellules de plus petites tailles (la taille de ces cellules est le 1^{er} paramètre). Nous comptons le nombre de fixations contenues dans chaque cellule et nous ne gardons que les cellules comprenant un certain pourcentage du nombre

total de fixations (le pourcentage est le 2^{e} paramètre).

La seconde étape consiste à rechercher l'enveloppe convexe de ces zones d'attention. Selon [GK99], l'enveloppe convexe des fixations d'un parcours oculaire fournit une mesure de la recherche effectuée par le sujet. Dans notre cas, dans la mesure où nous ne gardons que les zones ayant fait l'objet d'un certain seuil d'attention, cela fournit une mesure de la concentration de la recherche effectuée par le sujet. Si le sujet s'est concentré sur l'ensemble du diagramme, les fixations seront uniformément distribuées et le polygone sera grand. Dans le cas contraire, les fixations seront concentrées sur quelques zones de forte (selon le seuil) attention et le polygone sera plus petit.

Nous avons dès lors un polygone représentant la concentration de la recherche d'un sujet sur un diagramme. Ce polygone dépend de 2 paramètres : la taille des cellules et le seuil d'attention. Afin de déterminer la dissimilarité entre les différents polygones, nous utilisons la distance de Hausdorff qui prend en compte la forme des polygones et leur emplacement (section 4.3.1). Nous pouvons dès lors construire une matrice de dissimilarité comparant les 20 sujets pour 1 diagramme en fonction de 2 paramètres.

Il faut maintenant déterminer les paramètres à utiliser. Ces paramètres sont importants car ils produisent des matrices de dissimilarité différentes qui, suite à l'exécution d'AGNES, produiront des groupes non seulement de tailles différentes mais aussi de compositions différentes :

Taille : nous voulons classer nos 20 sujets en 2 groupes. L'application d'AGNES sur une matrice de dissimilarité fournira 2 groupes de tailles respectives : 1 et 19 ou 2 et 18 ou 3 et 17... 10 et 10.

Composition : si l'on prend k pixels comme paramètre pour les cellules et $x\%$ comme pourcentage de fixation pour les seuils, nous pouvons par exemple obtenir 2 groupes de tailles 5 et 15. Si l'on prend k' pixels comme paramètre pour les cellules et $x\%$ commme pourcentage de fixation pour les seuils, nous pouvons également avoir 2 groupes composés respectivement de 5 sujets et 15 sujets, sans pour autant que la composition des 2 paires de groupes soit la même.

Le choix des paramètres à utiliser devient dès lors subjectif et est sujet à biais. L'expérimentateur peut choisir les paramètres de manière à ce que cela lui donne les 2 groupes qui lui permettront de montrer ce qu'il a envie. Ce n'est pas une situation satisfaisante. Plutôt que de choisir subjectivement les paramètres à utiliser, nous avons fait le choix de générer l'ensemble des groupements possibles pour plusieurs valeurs données aux paramètres. Nous avons fait varier les paramètres selon le schéma suivant :

- la taille des cellules : 2^p pixels avec $p = 2, 3, \ldots, 8$.
- le pourcentage de fixation : de 0.1% à 10% avec un incrément de 0.1% à chaque palier.

Nous obtenons de cette manière : 24 combinaisons de paramètres produisant une matrice de dissimilarité fournissant un 1^{er} groupe de 1 sujet et un 2^{e} groupe de 19 sujets pour le diagramme sur JUnit ; 9 combinaisons de paramètres produisant une matrice de dissimilarité fournissant un 1^{er} groupe de 2 sujets et un 2^{e} groupe de 18 sujets pour le diagramme sur JUnit ; etc. Nous avons fait de même pour les 2 autres diagrammes. Les résultats sont présentés dans le tableau 8.2.

Nous pourrions sélectionner pour chaque diagramme les paramètres permettant de classifier les sujets en 2 groupes équilibrés, c'est-à-dire en 2 groupes de tailles $10 - 10$ (dernière ligne du tableau 8.2), mais comme nous l'avons dit, si les différentes combinaisons de paramètres donnent des groupes de taille équivalente $(10 - 10)$, les compositions de ces groupes sont différentes. Le choix serait dès lors également subjectif. De plus, les paramètres permettant d'obtenir 2 groupes de taille 10 pour le diagramme JUnit ne sont pas les mêmes que pour le diagramme QuickUML, ni ArgoUML.

Nous réduisons le problème en calculant à partir de ces groupes une seconde mesure de dissimilarité (expliquée ci-dessous). Prenons un exemple pour illustrer le procédé. Le tableau 8.2 montre qu'il y a 5 combinaisons de paramètres donnant naissance à des groupes de tailles 5 et 15 pour le diagramme JUnit. Nous prenons les 10 groupes (5 groupes de tailles 5 et 5 groupes de tailles 15)

Nombre de sujets dans chacun des 2 groupes	Nombre de combinaisons de paramètres possibles		
	JUnit	ArgoUML	QuickUML
$1-19$	24	21	38
$2-18$	9	14	17
$3-17$	13	12	5
$4-16$	10	6	1
$5-15$	5	8	8
$6-14$	4	10	1
$7-13$	9	2	1
$8-12$	2	2	0
$9-11$	7	4	1
$10-10$	2	5	1

TAB. 8.2: Récapitulatif du nombre de combinaisons de paramètres permettant de classer les sujets selon des groupes de tailles données.

pour calculer une matrice de dissimilarité selon la méthode décrite ci-dessous. Cette matrice de dissimilarité sera fournie à AGNES afin de calculer une nouvelle fois 2 groupes. Ces 2 groupes obtenus sont une agrégation des groupes précédents. Le calcul de la matrice de dissimilartité est expliqué ci-dessous.

Nous utilisons une mesure de similarité à partir des présences conjointes des sujets dans un groupe. Cette mesure est calculée à partir des fréquences de présence conjointe dans un groupe. C'est-à-dire que pour chaque paire de sujets, nous comptons le nombre de fois qu'ils sont présents conjointement dans un groupe. Plus le nombre obtenu est élevé, plus ils sont souvent présents en même temps dans un groupe et donc plus ils sont similaires. Une matrice de dissimilarité est remplie en prenant l'inverse du nombre calculé. Nous utilisons alors AGNES avec la matrice de dissimilarité obtenue. Si nous reprenons l'exemple du paragraphe précédent, les 5 groupes de taille 5 et les 5 groupes de taille 15 obtenus pour JUnit donnent, après agrégation, 2 groupes respectivement de taille 4 et 16. En utilisant le même procédé pour les autres diagrammes et pour chaque type de regroupement ($1-19$, $2-18$,..., $10-10$) nous obtenons 30 paires de groupes dont le tableau 8.3 reprend les tailles respectives.

Tailles des groupes d'origine	Taille des groupes obtenus		
	JUnit	ArgoUML	QuickUML
$1-19$	$1-19$	$2-19$	$1-19$
$2-18$	$2-18$	$2-18$	$1-19$
$3-17$	$1-19$	$3-17$	$2-18$
$4-16$	$4-16$	$4-16$	$4-16$
$5-15$	$4-16$	$5-15$	$5-15$
$6-14$	$5-15$	$6-14$	$6-14$
$7-13$	$7-13$	$7-13$	$7-13$
$8-12$	$8-12$	$8-12$	-
$9-11$	$9-11$	$9-11$	$9-11$
$10-10$	$10-10$	$10-10$	$10-10$

TAB. 8.3: Récapitulatif des tailles des groupes obtenus après agrégation.

Afin d'obtenir un groupement récapitulatif prenant en compte les 30 groupements différents du tableau 8.3, nous utilisons la même mesure de similarité (en prenant l'inverse nous avons une mesure de dissimilarité) pour construire une matrice de dissimilarité. La mesure prenant en compte 10 groupements par diagramme, ceux-ci devraient avoir la même influence sur le groupement final. En pratique, nous n'avons pu construire de groupement de tailles $8-12$ pour le diagramme sur

QuickUML (sigle '-' dans le tableau 8.3), ce qui signifie que ce diagramme a un poids légèrement inférieur.

Finalement, nous obtenons un groupement formé de 2 groupes de tailles $7 - 13$. Ce groupement est utilisé pour les analyses ultérieures. L'appartenance à un groupe est donnée par l'attribut « Cluster » dans le premier tableau de l'annexe A.1. Le sujet ayant la valeur 0 est le sujet retiré de l'ensemble, suite à la réduction de l'ensemble des données expliquée dans la section 8.2.

8.4 Étude des hypothèses principales

Le tableau 8.4 donne un résumé des types de réponses des 2 groupes déterminés à la section 8.3. Le groupe 1 est composé de 7 sujets et le groupe 2 de 13.

L'analyse du tableau 8.4 montre que le groupe 2 a un pourcentage de réponses sémantiquement correctes de près de 25% supérieur à celui du groupe 1 et un pourcentage de réponses moyennes de près de 7 % supérieur à celui du groupe 1. Malgré le taux de réponses fausses légèrement supérieur pour le groupe 2, la différence entre les réponses sémantiquement correctes est telle que *nous considérons que le groupe d'experts est représenté par le groupe 2 et que le groupe de novices est représenté par le groupe 1.*

Type de réponses	Groupe 1	Groupe 2
Réponses moyennes	45.24%	50%
Réponses sémantiquement correctes	9.52%	33.33%
Réponses sémantiquement non correctes	71.43%	33.33%
Réponses fausses	19.05%	33.33%

TAB. 8.4: Qualité des réponses des 2 groupes.

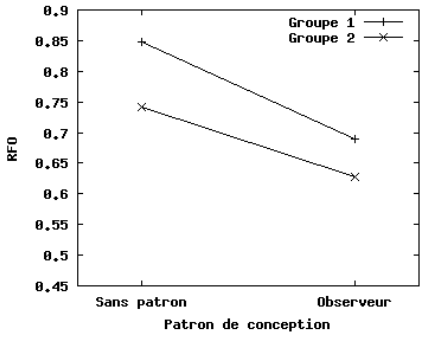
Rappelons que nous désirons mesurer la différence de performance entre les groupes et pas entre les diagrammes. L'expérience n'a pas été conçue pour permettre la comparaison entre les diagrammes. Les lignes qui nous intéressent dans les tables d'Anova sont donc celles en rapport avec les variables indépendantes : Expertise, Sommeil et Statut, ainsi que les lignes donnant les valeurs relatives aux interractions.

8.4.1 Le patron de conception Observateur

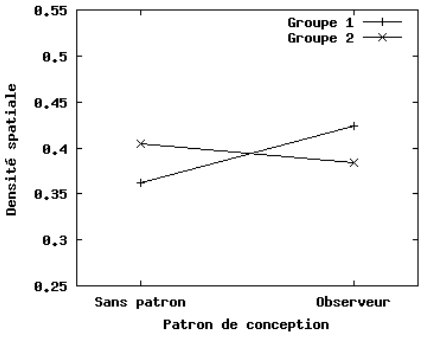
La figure 8.2a montre l'effet induit par la présence du patron de conception Observateur sur la mesure RFO. Le groupe de novices a une valeur plus élevée pour cette mesure, ce qui semble indiquer qu'il effectue un effort plus pertinent que le groupe d'experts. Néanmoins, le taux de réponses correctes est de manière générale plus élevé pour le groupe d'experts que pour le groupe de novices. La p-valeur (tableau 8.5) de la variable indépendante Expertise est de 0.0390. Elle est sous le seuil de 0.05, ce qui signifie que cette différence de pertinence de l'effort représentée par la mesure RFO est confirmée statistiquement. Il ne s'agit pas du résultat auquel nous nous attendions.

La variable indépendante Patron de conception semble avoir un effet sur la mesure RFO : les 2 lignes sont inclinées de la même manière. Les valeurs de cette mesure sont moins élevées pour le diagramme avec le patron Observateur. Cela signifie que les 2 groupes produisent un effort moins pertinent avec ce diagramme. Ici, la p-valeur est de 0.0015, c'est-à-dire qu'elle est sous le seuil de 0.05, ce qui indique une évidence statistique de l'influence de cette variable.

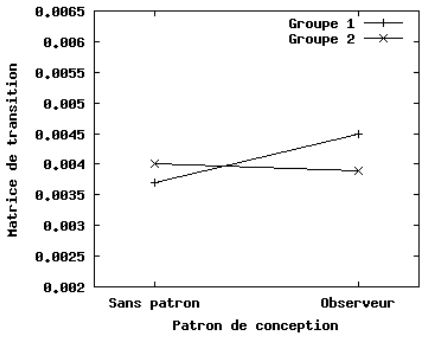
Toujours pour la mesure RFO, les 2 lignes sont relativement parallèles, ce qui laisse supposer qu'il n'y a pas d'interaction entre les 2 variables indépendantes et que les 2 groupes réagissent de la



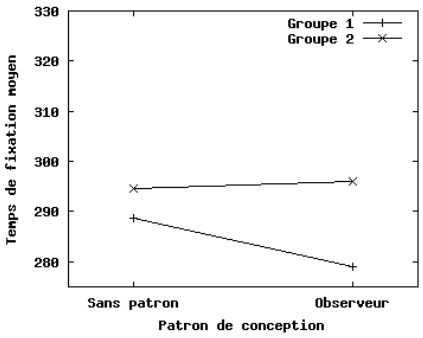
(a) Ratio de fixation



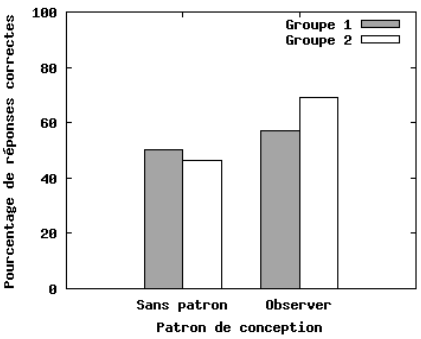
(b) Densité spatiale



(c) Matrice de transition



(d) Temps de fixation moyen



(e) Réponses correctes

FIG. 8.2: Impact du patron de conception Observateur pour les 2 groupes.

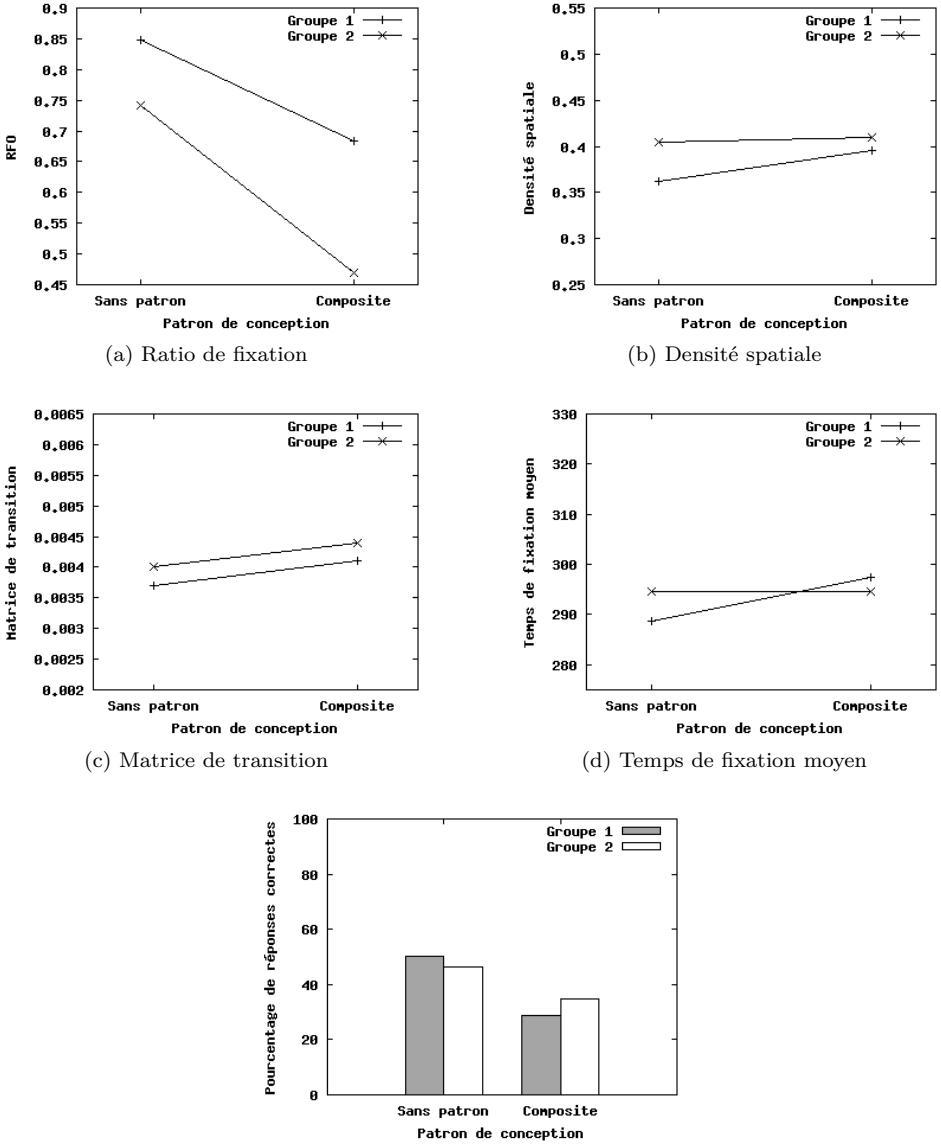
même manière à la présence du patron de conception. La p-valeur est de 0.5848, ce qui confirme notre analyse.

Facteur	RFO	DS	MT	TFM
Expertise	**0.0390**	0.9602	0.8016	0.5195
Patron de conception	**0.0015**	0.8423	0.7722	0.8882
Interaction	0.5848	0.3503	0.4931	0.7563

TAB. 8.5: Table d'Anova concernant l'impact du patron de conception Observateur sur les 2 groupes.

Une analyse du tableau 8.5 montre qu'aucune autre p-valeur n'est significative et que donc, seule la variable RFO semble être impactée. En particulier, aucune p-valeur relative à l'interaction des 2 variables indépendantes n'est significative, *ce qui ne permet pas de rejeter l'hypothèse nulle* H_{0_1}.

8.4.2 Le patron de conception Objet composite



(a) Ratio de fixation

(b) Densité spatiale

(c) Matrice de transition

(d) Temps de fixation moyen

(e) Réponses correctes

FIG. 8.3: Impact du patron de conception Objet composite pour les 2 groupes.

La figure 8.3a montre l'effet induit par la présence du patron de conception Objet composite sur la mesure RFO. Le groupe de novices a une valeur plus élevée pour cette mesure, ce qui semble indiquer qu'il effectue un effort plus pertinent que le groupe d'experts. De plus, le taux de réponses correctes est de manière générale égal entre les groupes. La p-valeur (tableau 8.6) de la variable indépendante Expertise est de 0.0005. Elle est sous le seuil de 0.05, ce qui signifie que cette

différence de pertinence de l'effort représentée par la mesure RFO est confirmée statistiquement. Il ne s'agit pas du résultat auquel nous nous attendions.

La variable indépendante Patron de conception semble avoir un effet sur la mesure RFO : les 2 lignes sont inclinées de la même manière. Les valeurs de cette mesure sont moins élevées pour le diagramme avec le patron Objet composite. Cela signifie que les 2 groupes produisent un effort moins pertinent avec ce diagramme. Ici, la p-valeur est de $1.1e^{-5}$, c'est-à-dire sous le seuil de 0.05, ce qui indique une évidence statistique de l'influence de cette variable.

Toujours pour la mesure RFO, les 2 lignes sont relativement parallèles, ce qui laisse supposer qu'il n'y a pas d'interaction entre les 2 variables indépendantes et que les 2 groupes réagissent de la même manière à la présence du patron de conception. La p-valeur est de 0.2092, ce qui confirme notre analyse.

Facteur	RFO	DS	MT	TFM
Expertise	**0.0005**	0.5174	0.8758	0.9329
Patron de conception	$\mathbf{1.1e^{-5}}$	0.7200	0.5066	0.8579
Interaction	0.2092	0.7293	0.8412	0.8048

TAB. 8.6: Table d'Anova concernant l'impact du patron de conception Objet composite sur les 2 groupes.

Une analyse du tableau 8.6 montre qu'aucune autre p-valeur n'est significative et que donc, seule la variable RFO semble être impactée. En particulier, aucune p-valeur relative à l'interaction des 2 variables indépendantes n'est significative, *ce qui ne permet pas de rejeter l'hypothèse nulle* H_{0_2}.

8.5 Analyse de l'impact des facteurs confondants

Pour analyser l'impact des facteurs confondants (le sommeil et le statut), il faudrait effectuer un test 3-way d'Anova. Nous n'avons pas effectué ce test car les 2 hypothèses nulles posées n'ont pas été réfutées et les interactions des 2 variables indépendantes ne semblent pas statistiquement significatives.

De plus, à notre connaissance, la taille minimale pour effectuer un test de normalité est de 4. Le groupe 1 composé de 7 sujets ne permet pas la création de 2 groupes de minimum cette taille. Ceci nous empêche d'effectuer le test de normalité, condition nécessaire à l'exécution du test du 3-way Anova.

Nous analysons tout de même les variables confondantes à l'aide du 2-way Anova afin de déterminer si ces variables ont une influence, ce qui nous permettra de mieux comprendre l'activité de compréhension logicielle.

Contrairement à l'analyse précédente, nous avons effectué les tests sur les 3 traitements en une fois.

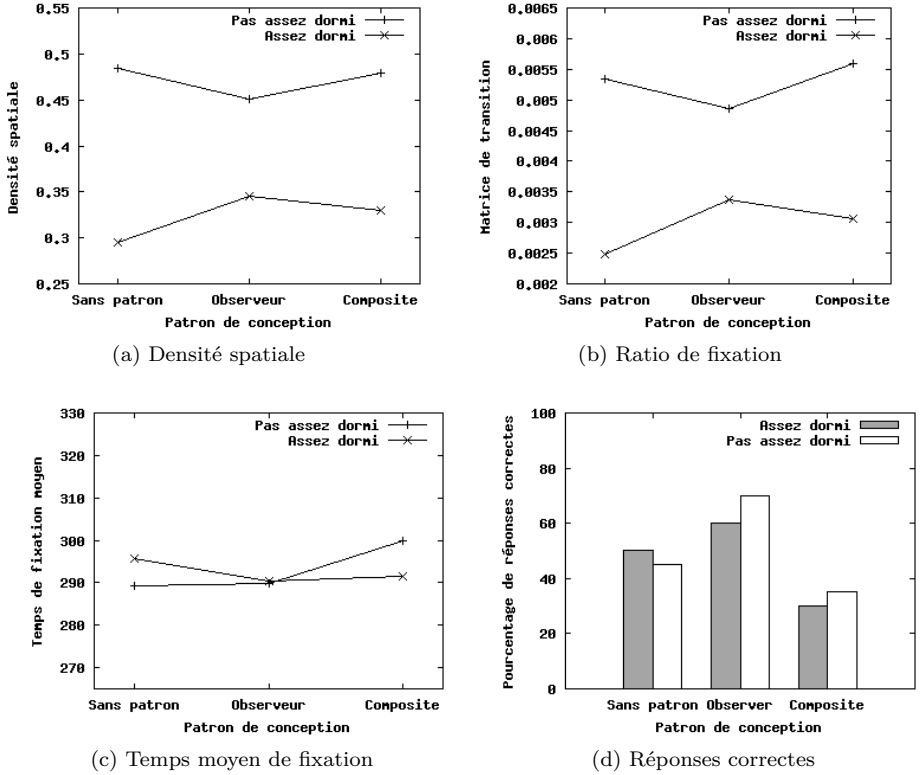
8.5.1 Analyse de l'impact du sommeil

À partir des informations récoltées concernant le sommeil des sujets, nous créons 2 groupes. Dans un groupe, nous plaçons les sujets ayant passé une nuit plus courte qu'à leur habitude. Dans l'autre groupe, nous plaçons les sujets ayant passé une nuit au moins aussi longue qu'à leur habitude. Cela nous donne 2 groupes équilibrés de 10 sujets.

Les hypothèses nulles que ce test tente de réfuter sont les suivantes :

- H_{0_3} : le sommeil a un impact sur l'effort moyen nécessaire pour effectuer une opération de maintenance sur un diagramme UML.
- H_{0_4} : l'impact d'un patron de conception sur l'effort moyen nécessaire pour effectuer une opération de maintenance sur un diagramme de classes UML est le même pour un groupe de sujets ayant assez dormi que pour un groupe de dujets n'ayant pas assez dormi.

Le test n'est pas effectué pour la variable RFO, l'un des groupes ne disposant pas d'une population normale. Les transformations de données habituellement d'usage n'ont pas permis la normalisation de la population. La variable RFO est donc absente de l'analyse.



(a) Densité spatiale (b) Ratio de fixation

(c) Temps moyen de fixation (d) Réponses correctes

FIG. 8.4: Impact du sommeil.

L'analyse de la figure 8.4d montre que la qualité des réponses des 2 groupes est globalement équivalente même si celle du groupe n'ayant pas assez dormi est très légèrement supérieure.

Les figures 8.4a et 8.4b, qui illustrent l'effort de recherche, montrent que le groupe n'ayant pas assez dormi effectue un plus gros effort de recherche. Les p-valeurs des 2 variables illustrées (densité spatiale et matrice de transition) sont respectivement de $7.2e^{-5}$ et 0.0002. Elles sont donc significatives et il est envisageable, sous réserve de la validité de l'expérience étudiée dans le chapitre 9, de réfuter l'hypothèse H_{0_3}. Le sommeil semble donc avoir un impact sur la quantité d'effort de recherche nécessaire à une tâche de maintenance sur un diagramme UML. Cette conclusion ne concerne que l'effort de recherche pour lequel les 2 variables dépendantes choisies sont statistiquement significatives. En ce qui concerne l'effort cognitif, il est impossible de tirer une conclusion.

Facteur	RFO	DS	MT	TFM
Patron	-	0.8036	0.7221	0.8889
Sommeil	-	$\mathbf{7.2e^{-5}}$	**0.0002**	0.8426
Interaction	-	0.2153	0.1954	0.8677

TAB. 8.7: Table d'Anova concernant l'impact du sommeil.

8.5.2 Analyse de l'impact du statut

Deux groupes ont été simplement formés sur base du statut des sujets. Un groupe est composé des professionnels ayant pris part à l'expérience, l'autre est formé des étudiants ayant pris part à l'expérience.

Les hypothèses nulles que ce test va tenter de réfuter sont les suivantes :
- H_{0_5} : l'effort moyen nécessaire à un groupe de professionnels pour effectuer une opération de maintenance sur un diagramme UML est égal à l'effort moyen nécessaire à un groupe d'étudiants.
- H_{0_6} : l'impact d'un patron de conception sur l'effort moyen nécessaire pour effectuer une opération de maintenance sur un diagramme de classes UML est le même pour un groupe de sujets professionnels que pour un groupe de sujets étudiants.

L'analyse de la figure 8.5e montre que la qualité des réponses des professionnels est globalement supérieure à celle des étudiants.

La figure 8.5a, qui illustre la variable RFO, montre que la variable indépendante Patron de conception a une influence. L'effort est globalement moins pertinent en présence d'un patron de conception. La p-valeur est de 0.0024. Elle est donc significative.

La figure 8.5d, qui illustre le temps de fixation moyen, montre que les professionnels effectuent un plus gros effort cognitif. La p-valeur concernant la variable indépendante Statut est de 0.0380. Elle est donc significative. Malheureusement, la seconde mesure (RFO) de l'effort cognitif ne confirme pas ce résultat. *Nous ne rejetons dès lors pas l'hypothèse nulle* H_{0_5}. Néanmoins, il ne semble pas « contre-intuitif » que des professionnels soient plus minutieux que des étudiants.

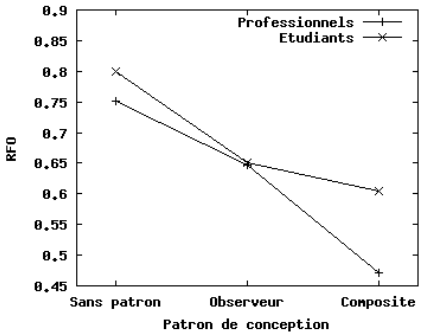
Facteur	RFO	DS	MT	TFM
Patron	**0.0024**	0.8432	0.7225	0.8821
Statut	0.3829	0.5106	0.8564	**0.0380**
Interaction	0.9677	0.8660	0.9085	0.9064

TAB. 8.8: Table d'Anova concernant l'impact du statut.

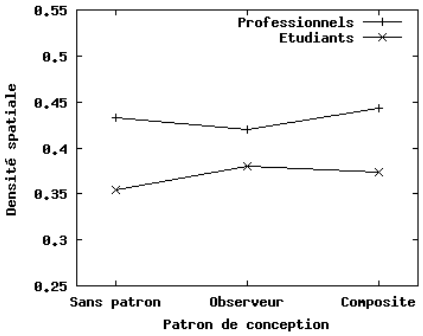
8.6 Conclusion de l'analyse

L'expérience réalisée n'a pas permis de réfuter les hypothèses nulles. De plus, il semblerait que le groupe de sujets experts produise un effort moins pertinent que le groupe de sujets novices, non seulement sur les diagrammes avec un patron de conception, mais également sur le diagramme sans patron de conception.

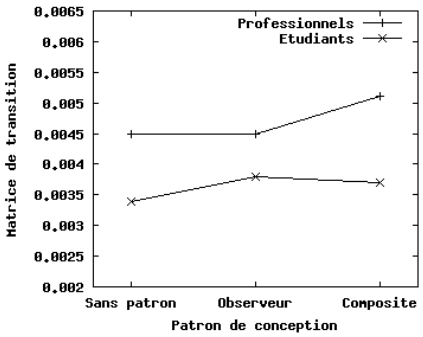
Si le groupe d'experts produisait un effort moins pertinent uniquement sur les diagrammes avec un patron de conception, nous pourrions supposer que les experts comprennent plus rapidement que les novices les fonctionnalités liées aux patrons de conception, mais qu'ils consultent tout de même le reste du diagramme pour s'assurer de leurs réponses, ce qui appuyerait la théorie « Vision-Compréhension ». D'un autre côté, s'ils ont compris correctement les fonctionnalités liées aux patrons de conception, ils ne devraient pas avoir à se rassurer en consultant le reste



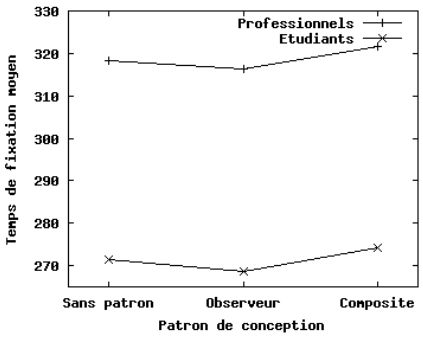
(a) Ratio de fixation



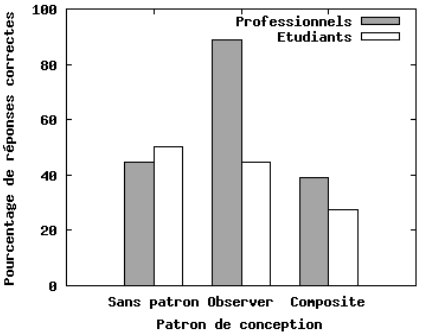
(b) Densité spatiale



(c) Matrice de transition



(d) Temps de fixation moyen



(e) Réponses correctes

FIG. 8.5: Impact du statut.

du diagramme. De plus, le groupe d'experts produit un effort moins pertinent sur le diagramme sans patron de conception également. Ces considérations nous font penser que les experts ont plus tendance à s'assurer de leurs réponses tandis que les novices donnent une réponse moins réfléchie. Cela semble se confirmer lorsque l'on analyse les réponses pour le diagramme sans patron. Contrairement aux experts, aucun sujet novice n'a donné de réponse sémantiquement correcte, alors que l'effort des novices est plus pertinent. De plus, l'effort de recherche ne semble pas différent dans les groupes. Cela semble confirmer que les experts, en passant moins de temps dans les zones pertinentes, en comprennent mieux les fonctionnalités, mais cela ne semble pas influencé par les patrons de conception.

Ces résultats vont donc à l'encontre de l'application de la théorie « Vision-Compréhension » aux patrons de conception. Il ne semble pas que les patrons de conception améliorent la compréhension

logicielle pour les experts. Nous n'avons cependant pas montré statistiquement que la théorie est fausse. Nous n'avons simplement pas réussi à vérifier son application aux patrons de conception.

De plus, nous classons les sujets selon leur examen des *stimuli*. Le fait que la variable RFO produise des résultats statistiquement significatifs suggère que notre but a été atteint : classer les sujets selon la manière dont ils examinent les *stimuli* et ensuite regarder si un groupe correspond plus à un groupe d'experts et l'autre à un groupe de novices. Cependant, si le taux de réponses sémantiquement correctes est fort différent entre les groupes, les moyennes le sont moins. De réels experts ne peuvent dès lors probablement pas être regroupés selon cette méthode.

L'expérience montre également que les sujets moins fatigués effectuent un effort de recherche inférieur à celui des sujets plus fatigués. Étant donné que le taux de réponses n'est pas réellement différent (46,7% pour les sujets moins fatigués et 50% pour les sujets plus fatigués, avec un léger avantage an niveau des réponses sémantiquement correctes pour les sujets moins fatigués), il semble que les sujets plus fatigués doivent effectuer un effort de recherche plus important pour situer l'information utile. Il semble donc que ce facteur soit bien un facteur confondant qu'il est sage de prendre en compte lors de la conception d'une expérience similaire à la nôtre. Nous pouvons faire 3 suppositions. Soit l'effort supplémentaire se fait sur des zones non pertinentes, soit l'effort supplémentaire se fait sur des zones pertinentes, soit l'effort supplémentaire se fait sur l'ensemble du diagramme. Nous regrettons de ne pas avoir pu étudier l'effet de cette variable sur la mesure RFO [2], ce qui aurait pu nous éclairer sur la pertinence de l'effort de recherche supplémentaire déployé par les sujets fatigués.

Il nous semble, vu le nombre inconnu de facteurs confondants potentiels, que l'utilisation que nous avons voulu faire de l'oculométrie est légèrement prématurée. Il faut, selon nous, récolter plus de connaissances basiques concernant l'utilisation que fait un programmeur des diagrammes de classes UML. La différence entre l'effort de recherche des personnes ayant assez dormi par rapport à celles n'ayant pas assez dormi en est l'illustration. Ce travail permettrait de connaître les facteurs confondants à neutraliser pour concevoir des expériences plus solides.

La section suivante propose une analyse de la validité de l'expérience, précieuse pour le chercheur voulant reproduire et améliorer l'expérience.

[2] Pour rappel, les échantillons ne respectaient pas la loi normale.

Chapitre 9

Menaces à la validité de l'expérience

La validité des résultats obtenus par une expérimentation fait l'objet de plusieurs menaces. Selon [WRH+00], il existe 4 types de validités pour une expérience : la validité de conclusion, de construction, interne et externe. Ces différents types de validités sont parfois exclusifs et sont présentés dans la section 3.2.4. Le but de l'expérimentateur est alors de mitiger les menaces ou de les accepter en fonction du but de son expérimentation. Une liste de menaces est proposée par [WRH+00] et selon eux, elle peut servir de « checklist » à leur détection durant la conception d'une expérience. Ce chapitre passe en revue chacune des menaces et explique comment elles ont été mitigées ou acceptées.

9.1 Validité de conclusion

Faible puissance statistique. Soit une hypothèse nulle, la puissance statistique d'un test est la probabilité qu'il rejette cette hypothèse lorsqu'elle est fausse. Cette probabilité dépend de :
- la différence réelle de performance entre les populations ;
- la variabilité intra-population (mesurée par l'écart-type) ;
- le seuil de probabilité α ;
- la taille de l'échantillonnage.

Les seuls éléments permettant d'avoir une influence sur cette puissance sont le seuil de probabilité, qu'il n'est pas judicieux de modifier, et la taille de l'échantillon qui n'est pas simple à construire. Afin de limiter cette menace, le nombre de diagrammes intervenant dans l'expérience est limité au strict minimum de manière à ce que tous les sujets interviennent sur chaque diagramme.

Violation des hypothèses de test statistique. Afin d'utiliser les tests d'hypothèses paramétriques, les données doivent respecter certaines hypothèses. Les 2 principales sont la normalité de la population et l'indépendance des échantillons.

Le test du 2-way Anova que nous avons utilisé demande également l'homogénéité des variances. La normalité de la population et l'homogénéité de la variance ont fait l'objet d'un test. Les résultats de ces tests sont disponibles dans l'annexe A. Dans le cas où les populations n'étaient pas normales, les tests étaient effectués après une transformation logarithmique des données. De cette manière, seulement un test (concernant l'analyse du sommeil, section 8.5.1) n'a pu être réalisé à cause de cette contrainte et tous les autres tests la respectent. L'indépendance des échantillons fait référence au fait que les réponses d'un sujet ne sont pas influencées par celle d'un autre sujet. Pour cette partie, la confiance dans le sérieux des sujets est de mise et la menace est acceptée.

« Fishing » et taux d'erreur. L'expérimentateur se doit de rechercher un résultat spécifique. Récolter des données et effectuer tous les tests possibles et imaginables dessus afin de trouver « quelque chose » n'est pas une bonne méthode. Le problème est que l'expérience n'a pas été conçue pour la recherche de ce « résultat », si on peut l'appeler ainsi, et beaucoup de facteurs externes sont susceptibles d'être la cause de ce « résultat ». De même, la recherche d'un résultat trop spécifique est aussi une menace car l'analyste n'est plus objectif.

Afin de mitiger cette menace, la planification préalable de l'expérience précise clairement les phénomènes observés. De plus, les mesures choisies sont issues de la littérature et sont sélectionnées lors de cette planification, de manière à éviter leur (re)définition et à garder une objectivité.

Fiabilité des mesures. La validité d'une expérience dépend fortement de la fiabilité des mesures. Cela peut dépendre de différents facteurs tels qu'une mauvaise instrumentation. Le principe de base est que, lorsqu'on mesure un phénomène 2 fois, le résultat est identique.

Dans ce cas-ci, les mesures sont effectuées automatiquement via l'oculomètre. Dès lors qu'aucune intervention humaine ne se fait, le résultat (pour un sujet se comportant exactement de la même manière) sera identique si la mesure est répliquée et pour peu que la calibration de l'appareil aie bien été effectuée. La plus grande attention a donc été portée lors de cette calibration.

La fiabilité des données comme représentant le suivi du regard du sujet n'est pas toujours évidente. Les mouvements de la tête ont un impact considérable sur les données récoltées et sont très gênants. Une correction partielle des données a donc été effectuée avant leur analyse. Cette correction a été décrite dans la section 7.3.2.

Une autre menace est l'évaluation des réponses données par les sujets. Nous avons analysé la documentation fournie avec les systèmes (code, diagrammes, etc.) afin d'évaluer les réponses, néanmoins, nous ne les avons pas implémentées concrètement afin de vérifier leur justesse. Même s'il nous semble que l'évaluation s'est effectuée minitieusement, cela reste une menace. Nous l'acceptons.

Fiabilité de l'implémentation des traitements. Il s'agit du risque que l'implémentation soit différente en fonction des sujets. Chaque sujet recevant les mêmes traitements pour notre expérience (section 7.2.5), cette menace ne devrait pas peser sur notre expérience.

Non pertinence aléatoire dans les paramètres de l'expérience. Des éléments externes aux traitements et facteurs peuvent influencer le résultat (bruit, fatigue, etc.). Pour mitiger cette menace, la pièce où se déroulait l'expérience était calme et un panneau demandant de ne pas être dérangé était affiché. La fatigue est un élément analysé dans la section 8.5. Enfin, une attention particulière a été portée à ce que le sujet soit dans les meilleures conditions pour passer l'expérience : pas de surprise et tentative d'élimination du stress.

Hétérogénéité aléatoire des sujets. Dans une étude, le groupe de sujets est, dans une certaine mesure, toujours hétérogène. Si le groupe est fort hétérogène, il y a un risque que les variations soient dues à des différences individuelles plutôt qu'aux traitements. D'un autre côté, choisir un groupe homogène aura un impact sur la validité externe de l'expérience.

Dans notre expérience, il s'agit justement de confronter 2 groupes, avec un bagage homogène au sein de chaque groupe, afin de déterminer l'impact de ce bagage. Ce sont donc les ressemblances qui nous intéressent et non les différences. La menace concernant la validité externe est acceptée.

9.2 Validité interne

Il existe 3 types de menaces à la validité interne d'une expérience.

9.2.1 Menaces sur un groupe

Les menaces sur un groupe affectent aussi les expériences avec plusieurs groupes. Une expérience comme celle-ci, avec 2 groupes, est donc sujette à ces menaces.

Historique. Lors d'une expérience, les traitements sont appliqués aux objets à différents moments. Les circonstances n'étant pas les mêmes, cet historique risque d'influencer les résultats. Cela peut être le cas si l'expérience a lieu le premier jour après des vacances ou un jour où un événement particulier a eu lieu.

Il s'agit ici d'un risque particulièrement difficile à limiter. Qu'est-il possible de faire pour éviter l'influence d'une mauvaise interrogation portant sur l'UML par exemple ? C'est une menace qu'il faut accepter. Néanmoins, quelques recommandations étaient fournies aux sujets lors de leur inscription à l'expérience (*e.g.*, ne pas se présenter avec un manque de sommeil ou de ne pas consommer la veille des substances pouvant avoir un effet prolongé, tel l'alcool).

Maturation. Ce sont les différentes manières dont les sujets réagissent au temps. Il peut s'agir d'une réaction qui affecte négativement (fatigue ou ennui) ou positivement (apprentissage). Cette menace est particulièrement pertinente pour cette expérience. La fatigue des sujets les rend non seulement enclins à effectuer des mouvements de tête, mal supportés par l'oculomètre, mais également moins performants.

Pour limiter le risque de mouvement dû à la fatigue, l'expérience a été conçue pour être la plus courte possible. Pour limiter l'effet d'apprentissage, les diagrammes étaient issus de systèmes différents. De plus, les tâches étaient présentées aux sujets dans un ordre aléatoire, ce qui mitige l'effet de la fatigue et celui de l'apprentissage dans les données récoltées.

Essais. La répétition de tests peut faire en sorte que les sujets répondent différemment parce qu'ils apprennent la manière avec laquelle ces tests sont conduits. Pour limiter cette menace, une mise en situation était effectuée afin que chaque sujet sache exactement la manière dont l'expérience se déroule (section 7.3.1). Le but étant d'éviter toute surprise et que chaque sujet ait la même préparation.

Instrumentation. Il s'agit de l'effet causé par les objets utilisés pour l'exécution de l'expérience. S'ils sont mal définis, l'expérience en sera affectée négativement. De manière générale, la section 7.3.1 sur l'instrumentation traite de cette menace. Cette menace fait également encore référence au biais, dans les données enregistrées, dû aux mouvements de tête du sujet. Il existe également un risque de mauvaise appréciation des zones pertinentes des diagrammes. Les zones pertinentes ont été définies comme les zones renfermant une information nécessaire à la formulation de la réponse et ont été analysées avec l'aide du professeur Yann-Gaël Guéhéneuc.

Une menace concerne le diagramme comprenant le patron de conception Objet composite qui renferme une erreur (figure 9.1). Une relation d'héritage devant être représentée par une flèche est représentée comme une agrégation. Selon [Gué06], les relations interclasses ne sont pas fort utilisées par les programmeurs dans leurs tâches et nous espérons que cette erreur n'a pas de conséquence significative. Néanmoins, il s'agit d'une menace.

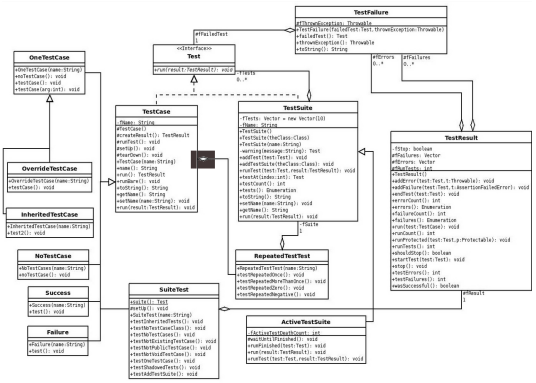


FIG. 9.1: Le diagramme comprenant le patron de conception Objet composite.

Régression statistique. Cette menace se présente lorsque les sujets sont classés par groupes sur base d'une autre expérience ou d'une expérience passée. Il se peut qu'il y ait une modification qui ne soit due à aucun traitement. Imaginons que l'on prenne les 10 moins bons d'une expérience passée, ces 10 sujets ne peuvent que s'améliorer et les variations seront dues à des variations aléatoires et non aux traitements.

Notre expérience n'utilise pas ce type de procédé, la menace est donc écartée.

Sélection. Il s'agit de la variation naturelle de la performance humaine. La manière de sélectionner l'échantillon a un effet. Dans ce cas-ci, le fait de ne prendre que des volontaires comme sujets peut influencer les résultats. Les volontaires sont généralement plus motivés et mieux adaptés à de nouvelles tâches que le reste de la population. D'un autre côté, il n'est pas intéressant d'obliger quelqu'un à prendre part à l'expérience car on ne peut l'obliger à y prendre part avec le sérieux nécessaire. Le groupe sélectionné n'est donc pas représentatif de la population entière mais c'est donc une menace que nous sommes contraint d'accepter. Néanmoins, comme nous avons affaire à une expérience spécifique (voir la définition du contexte dans la section 7.2.1), cela ne pose pas de problème.

Mortalité. Cette menace est due aux sujets qui abandonnent l'expérience. Si tous les sujets représentatifs d'une catégorie abandonnent, les résultats de l'expérience seront biaisés. Dans notre cas, nous n'avons pas eu d'abandons excepté le sujet pour lequel nous ne sommes pas arrivés à calibrer l'oculomètre et le sujet que nous avons refusé à cause de son manque de connaissance de la notation UML. Cette menace n'a donc pas lieu d'être.

Ambiguïté sur l'influence de la cause. Est-ce que A cause B, B cause A ou encore X cause A et B ? C'est une menace très importante pour cette expérience. Elle apparaît à plusieurs moments.

La première question posée est celle de la validité des traitements. Est-ce bien ceux-ci qui sont à la base des phénomènes observés ? Nous faisons l'hypothèse qu'un diagramme renfermant un patron de conception est représentatif de ce patron de conception. Nous faisons également l'hypothèse que la comparaison des performances sur 2 diagrammes différents est valide. Or, les différences peuvent venir par exemple de la différence de complexité dans la tâche. Il est clair que ces hypothèses sont fortes mais les relâcher aurait comme influence de renforcer d'autres menaces. Par exemple, prendre les mêmes diagrammes légèrement modifiés renforcerait la menace de maturation. Afin de limiter cette menace, il faudrait ne pas appliquer tous les traitements à chaque sujet, ce qui renforcerait la menace de faible puissance statistique, etc. Il faut donc effectuer un choix et le nôtre a été d'accepter ces menaces.

La seconde ambiguïté peut survenir lors de la définition des groupes décrite à la section 8.3 Ces groupes sont formés sur base des données collectées et il est donc possible d'obtenir une dépendance cyclique. Les données définissent les groupes qui définissent les résultats. Afin de mitiger ce risque, le calcul des variables dépendantes est fait de manière bien différente du calcul des groupes. Une exception doit tout de même être mise pour la densité spatiale.

9.2.2 Menaces sur plusieurs groupes

Les menaces sur plusieurs groupes ne s'appliquent qu'en présence de plusieurs groupes.

Interaction avec la sélection. L'interaction avec la sélection est due à un comportement différent dans les différents groupes. Par exemple, certains groupes vont mûrir de manière plus rapide que d'autres. Lors de l'utilisation d'une nouvelle méthode, par exemple, certains groupes vont l'apprendre plus rapidement que d'autres. Dans le cas présent, il s'agit justement de mesurer les différences entre 2 groupes exécutant une tâche de compréhension d'un schéma. Cette menace est donc l'objet de l'expérience même. Elle peut donc être écartée.

9.2.3 Menaces sociales

Ce types de menaces s'appliquent qu'il y ait un ou plusieurs groupes.

Diffusion ou imitation de traitement. Cet aspect se présente lorsqu'un groupe obtient des informations concernant les traitements d'un autre groupe ou tente d'imiter le comportement d'un autre groupe. Cette menace peut également apparaître intra-groupe. Afin de mitiger ce risque, nous avons explicitement demandé aux sujets de ne dévoiler aucune information concernant le déroulement de l'expérience.

Équité des traitements compensatoires. Cette menace intervient lorsque les traitements diffèrent en fonction des groupes, ce qui n'est pas notre cas. Elle est donc acceptée.

Rivalité compensatoire. Un sujet recevant des traitements moins enviables peut être motivé à réduire ou inverser le résultat de l'expérience. Un groupe utilisant une méthode traditionnelle peut faire de son mieux afin de montrer la compétitivité de cette méthode. Dans le cas présent, chaque groupe reçoit le même traitement. Cette menace ne pèse donc pas sur l'expérience.

Ressentiment dû à la démoralisation. C'est l'opposé de la menace précédente. Un sujet recevant des traitements moins enviables peut abandonner et ne pas être aussi performant que d'habitude. Le groupe utilisant l'ancienne méthode peut ne pas être motivé pour réaliser un bon travail alors qu'apprendre quelque chose de nouveau inspire le groupe utilisant la nouvelle méthode. Comme pour le point précédent, cette menace n'a pas lieu d'être.

9.3 Validité de construction

9.3.1 Menaces de conception

L'insuffisance de la définition de la construction. Cela signifie simplement que l'expérience n'est pas suffisamment préparée, avant d'être traduite en mesures et traitements. La théorie n'est

pas assez claire, c'est pourquoi l'expérience ne peut être suffisamment claire. Prenons l'exemple de la comparaison de 2 théories sans avoir au préalable défini la signification de « meilleur ».

Cette menace est présente au niveau de nos définitions des mesures cognitives. Il n'existe pas de certitude quant à leur signification. Faute de mieux, cette menace est acceptée.

Biais mono-opérationnel. Dans le cas où l'expérience ne fait intervenir qu'une seule variable indépendante, qu'un seul sujet ou traitement, l'expérience peut sous-représenter la construction et donc ne pas donner une image assez représentative de la théorie. Cette menace est acceptée.

Nous pensons qu'elle est assez forte pour notre expérience. Non à cause du fait que nous n'ayons pris que peu de variables indépendantes en compte, mais à cause du fait que le domaine reste peu connu et que nous n'avons probablement pas fixé l'ensemble des variables indépendantes de confusion. La construction s'en trouve de ce fait sous-représentée.

Biais mono-méthode. N'utiliser qu'un seul type de mesure ou d'observation n'implique un risque que si la mesure ou l'observation est sujette à biais ; alors l'expérience est erronée. En impliquant différents types de mesures et observations, il est possible de croiser les résultats. Plusieurs variables dépendantes ont été prises en compte pour permettre le recoupement des résultats.

Confusion de la construction avec le niveau de construction. Dans certaines relations, ce n'est pas la présence ou l'absence d'une construction, mais le niveau de la construction qui est importante pour le résultat. L'effet de la présence de cette construction est confondu avec l'effet du *niveau* de la construction. Par exemple, ce n'est pas la présence de connaissance concernant les patrons de conception qui est la cause d'une relation mais le niveau de connaissance de ces patrons qui peut expliquer la différence. Pour notre expérience, nous ne considérons pas l'expérience en terme de niveau (section 7.2.6). Nous acceptons cette menace.

Interaction de différents traitements. Si le sujet est impliqué dans plus d'une étude, les traitements de plusieurs études peuvent interagir. Il est alors impossible de conclure que les effets sont dus à un traitement ou à une combinaison de traitements. Aucune expérience similaire à la nôtre n'a été réalisée durant l'exécution de la nôtre. De plus, la participation à l'expérience ne nécessitait qu'une présence unique. Cette menace ne semble donc pas pertinente.

Interaction des tests avec les traitements. Le test lui-même, l'application de traitements par exemple, peut rendre le sujet plus sensible ou réceptif au traitement. Le test fait alors partie des traitements. Par exemple, si le test fait intervenir la mesure du nombre d'erreurs faites en codant, alors le sujet sera plus attentif et va donc les réduire. Afin de réduire cette menace nous n'avons pas explicitement informé les sujets du but de l'expérience. Il s'agit d'une expérience en génie logiciel concernant la compréhension de diagrammes de classes. Il n'est nullement fait objet de patron de conception. Afin de ne pas éveiller les soupçons, nous ne donnons pas d'autres informations concernant le but précis de l'expérience.

Limitation de généralisation à travers la construction. Le traitement peut affecter positivement la construction étudiée mais peut également affecter non intentionnellement d'autres constructions. Cette menace rend les résultats difficiles à généraliser. Par exemple, une étude comparative conclut qu'une méthode augmente la productivité. D'un autre côté, cette méthode diminue la facilité de maintenance, ce qui est un effet de bord. Si la maintenabilité n'est pas mesurée ou observée, il y a un risque que la conclusion soit tirée de l'attribut productivité sans prendre en compte la maintenabilité.

Comme expliqué plus haut, le but n'est pas de généraliser outre mesure les résultats de cette expérience. Néanmoins, les réponses sont évaluées selon 2 angles. Les réponses correctes faisant une bonne utilisation de la sémantique du diagramme et les réponses correctes ne faisant pas une bonne utilisation de la sémantique du diagramme. Dans une tâche de maintenance ponctuelle, les réponses ne respectant pas la sémantique ne portent pas de préjudice mais avec le temps elle augmentent la charge de travail nécessaire pour la maintenance.

9.3.2 Menaces sociales

Hypothèse de deviné. Lorsque des personnes prennent part à une expérience, elles peuvent tenter de deviner le but de l'expérience. Ensuite, elles sont susceptibles de baser leur comportement sur cette supposition, ce qui influencera le résultat. Cette menace est très difficile à mitiger. Même si le sujet ne devine pas le but de l'expérience, il risque de baser son comportement sur le but qu'il imagine.

Nous avons donc, pour mitiger cette menace, bien expliqué le but de l'expérience mais sans entrer dans les détails, ne précisant pas qu'il porte sur les patrons de conception. De plus, les sujets pouvaient se retirer à tout moment sans aucune pénalité.

Appréhension d'évaluation. Certaines personnes ont peur d'être évaluées. Elles ont dès lors tendance à tenter de sembler meilleures, ce qui influence le résultat de l'expérience. Afin de limiter ce risque, nous rendons les résultats anonymes et le faisons savoir aux sujets afin qu'ils ne soient pas inquiétés. De plus, aucune limite de temps n'a été donnée afin de ne pas ajouter de pression inutile aux sujets.

Attentes de l'expérimentateur. Un expérimentateur peut biaiser les résultats d'une étude consciemment ou inconsciemment sur base de ce qu'il attend de l'expérience. La menace peut être réduite grâce à l'implication de différentes personnes n'ayant pas – ou d'autres – attentes de l'expérience. Dans notre cas, plusieurs personnes ont été impliquées afin de confirmer les choix relatifs à l'expérience. De plus, le fait de réfuter une hypothèse est également très bon pour l'avancement de la compréhension en génie logiciel. Il n'y a donc pas d'attentes particulières.

9.4 Validité externe

Interaction entre la sélection et les traitements. Cette menace concerne le fait d'avoir un échantillon non représentatif de la population pour laquelle nous voudrions généraliser les résultats. Cette menace est généralement forte car les sujets sont souvent des étudiants et la question de savoir si les étudiants forment une population représentative se pose. Dans ce cas, près de la moitié des sujets étaient des professionnels, ce qui affaiblit considérablement cette menace.

Interaction entre les paramètres et les traitements. Cette menace résulte du fait de ne pas disposer de matériel ou de paramètres expérimentaux représentatifs des pratiques industrielles par exemple. L'utilisation d'outils démodés pour l'expérience, alors que l'industrie utilise la pointe de la technologie, en est un exemple. L'utilisation de problèmes spécifiques est un autre exemple.

Étant donné que nous ne sommes qu'au début des expériences concernant la compréhension des patrons de conception, le but n'est pas encore de généraliser les résultats mais bien d'obtenir de petites conclusions valides dans certains contextes afin d'orienter les travaux futurs. Cette menace est donc acceptée.

Interaction entre l'histoire et les traitements. Que l'expérience soit conduite un jour spécial peut affecter le résultat. Le remplissage d'un questionnaire concernant un logiciel de sécurité critique quelques jours après que ce logiciel soit « tombé » n'est pas une bonne idée. Les gens auront tendance à répondre différemment que quelques jours auparavant. Cette menace est également acceptée car très difficile à mitiger.

Chapitre 10

Conclusion et travaux futurs

Conclusion

Ce mémoire a pour sujet, premièrement, l'étude des éléments et disciplines touchant à la réalisation d'une expérimentation en génie logiciel, utilisant la technique de l'oculométrie ainsi que des représentations sous forme de diagrammes de classes UML, et, deuxièmement, deuxièmement, le compte-rendu d'une expérience que nous avons réalisée pour tenter d'apporter un élément pouvant appuyer ou réfuter la théorie « Vision-Compréhension ».

Au chapitre 2, nous expliquons la théorie « Vision-Compréhension » qui, en joignant les théories de la science de la vision à des théories relatives à la compréhension logicielle, tente d'expliquer l'acquisition des informations par un programmeur impliqué dans une activité de compréhension logicielle.

Au chapitre 3, nous expliquons qu'une expérimentation se doit de contrôler *tous* les facteurs qui entrent en jeu. Les chapitres suivant présentent les éléments et disciplines qui touchent à la réalisation de notre expérience. Le nombre de ces éléments est révélateur de la difficulté à réaliser une expérience comme la nôtre tant il y a de facteurs à prendre en compte et ce, particulièrement dans un contexte où on désire tirer des conclusions sur la cognition. La nature interne des processus cognitifs fait qu'ils sont compliqués à étudier, particulièrement lorsque l'on ne peut qu'analyser les comportements externes.

L'utilisation de l'oculométrie pour la réalisation de notre expérience est tout indiquée. Malheureusement, cette discipline souffre d'un manque de maturité et la complexité de l'interprétation des variables dépendante est un sérieux frein. En effet, l'interprétation des variables peut varier en fonction du contexte. Néanmoins, son utilisation est un formidable moyen pour en apprendre davantage sur la manière dont les programmeurs utilisent les diagrammes de classes UML ou tout autre documentation graphique.

Nous avons utilisé l'oculométrie sur des diagrammes de classes UML afin de valider ou réfuter l'application de la théorie « Vision-Compréhension » aux patrons de conception Observateur et Objet composite. Il semble que le groupe d'experts, tout en passant moins de temps dans les zones pertinentes des diagrammes en comprennent mieux les fonctionnalités. Cependant, cela ne semble pas être influencé par les patrons de conception. Les résultats de l'expérience sont donc en défaveur de l'application proposée par l'auteur de la théorie aux patrons de conception. Il est à noter que nous n'avons pas montré que l'application de la théorie aux patrons de conception était fausse, ni que les patrons ne facilitent pas la compréhension. Nous n'avons simplement pas pu montrer statistiquement que c'était bien le cas.

Par contre, les résultats sont sans appel concernant un facteur confondant étudié : le sommeil. Il est statistiquement évident que des sujets fatigués fournissent un plus gros effort de recherche

que des sujets moins fatigués pour un même résultat. Ce constat est intéressant car il permet, et nous voyons ici l'intérêt des recherches empiriques, de trancher entre 2 théories possibles : « un sujet fatigué aura tendance à effectuer un plus petit effort de recherche » ou « un sujet fatigué aura tendance à effectuer un plus gros effort de recherche ». En effet, que ce soit l'une ou l'autre, il est possible de trouver une explication théorique : si les sujets plus fatigués faisaient moins d'efforts de recherche que les sujets moins fatigués, nous pourrions supposer qu'ils sont moins « courageux » ; mais comme les sujets plus fatigués font plus d'efforts de recherche que les sujets moins fatigués, nous pouvons supposer que le sommeil implique un supplément de charge cognitive pour la localisation de l'information utile. Malheureusement, nous n'avons pas pu étudier la nature de l'effort supplémentaire consenti [1] par les sujets plus fatigués. Nous ne savons donc pas si les efforts supplémentaires se font dans les zones pertinentes ou pas. Dans tous les cas de figure, il s'agit là d'un facteur à neutraliser pour les expériences futures.

Ce constat est représentatif de la difficulté, technique cette fois, de réaliser une expérience. Comme nous l'avons précisé, il est déjà compliqué de recruter un nombre suffisant de participants pour une telle expérience. Si, en plus, on désire qu'ils aient, par exemple, tous dormi suffisamment d'heures, le recrutement devient très compliqué, non seulement pour trouver les sujets acceptant de participer mais en plus pour s'assurer que la consigne est respectée. Plus on découvrira de facteurs confondants, plus une expérience sera compliquée à mettre en œuvre. Mais si on détermine que des facteurs que l'on pensait confondants ne le sont pas, cela permettra également d'alléger les mesures prises pour les contrôler. Un moyen théoriquement simple de compenser ces facteurs confondants et d'augmenter le nombre de sujet. Mais pratiquement, il est difficile de trouver des sujets motivés.

Selon nous, il faut donc impérativement soigneusement étudier le contexte d'utilisation avant d'avoir recours à l'oculométrie. De plus, il ne faut pas être trop ambitieux et tenter d'analyser des faits trop complexes dans un premier temps.

Travaux futurs

Selon nous, la validation ou la réfutation de la théorie « Vision-Compréhension » passe effectivement par l'oculométrie. Cette technique permet d'ouvrir la boîte noire du processus cognitif par l'observation de l'attention visuelle. Cependant, à l'image de la complexité des processus cognitifs, l'interprétation des données récoltées est complexe. La conception *ad-hoc* de mesures uniquement sur des considérations théoriques nous semble être une mauvaise approche et nécessite un modèle, surtout lorsque l'on utilise des *stimuli* et des tâches provoquant des comportements complexes.

Les travaux futurs devraient prioritairement s'atteler à définir un cadre d'interprétation des mesures en oculométrie dans le contexte du génie logiciel. Ce cadre d'interprétation devrait au moins tenir compte des spécificités des *stimuli* (*e.g.*, des diagrammes de classes UML) et de la diversité des tâches demandées aux sujets :

Stimulus : le processus cognitif de la compréhension d'un diagramme de classe, par exemple, nécessite la mise en relation des différents éléments du diagramme. La mise en relation des éléments implique un comportement plus complexe que celui étudié dans l'étude des interfaces Homme-Machine, par exemple. Des mesures adaptées devraient dès lors être conçues et surtout validées de manière empirique.

Tâche : la diversité des tâches est également à prendre en compte car elles induisent des processus cognitifs différents. Si on demande à un sujet de trouver quelle classe renferme un élément, son comportement sera différent de celui qu'il aura s'il doit mettre en relation cet élément avec d'autres. Par exemple, un changement brusque de zone d'intérêt aura une signification différente : si le sujet doit uniquement trouver un élément dans le diagramme, cela peut signifier qu'il ne l'a pas trouvé et qu'il va le chercher ailleurs ; si le sujet doit mettre en relation

[1] Pour rappel, les données ne respectent pas les hypothèses d'application du test du 2-way Anova.

des éléments, soit il n'a pas trouvé l'information qu'il cherche et il va la chercher ailleurs, soit le sujet a trouvé l'information qu'il cherche et se met à la recherche de l'information suivante. Ce sont des suppositions qu'il est nécessaire de valider empiriquement.

Selon nous, la méthode pour arriver à définir ce cadre général d'interprétation passe par l'isolation de l'ensemble des facteurs susceptibles d'avoir une influence sur la manière avec laquelle les sujets réagissent aux *stimuli*, afin de disposer de connaissances de base solides. En effet, aujourd'hui nous ne disposons (à notre connaissance) que de connaissances limitées : Guéhéneuc [Gué06] a montré que les sujets ont tendance à se concentrer sur les classes plutôt que sur les relations entre les classes ; Jeanmart [Jea08] a montré que des sujets avec une bonne connaissance d'UML ont tendance à se concentrer davantage sur les éléments comprenant un élément de réponse ; enfin nous avons montré que l'effort de recherche consenti est dépendant de l'état de fatigue. De telles connaissances permettront également de mieux neutraliser les facteurs confondants et donc de concevoir des expériences plus ambitieuses et solides. Le tout nous permettra de valider et de réfuter la théorie « Vision-Compréhension » et donc d'avancer dans l'activité de la compréhension de logiciels. Cette étape est indispensable pour améliorer cette activité et pourquoi pas l'automatiser partiellement.

Bibliographie

[ACC+07] Lerina Aversano, Gerardo Canfora, Luigi Cerulo, Concettina Del Grosso, and Massimiliano Di Penta. An empirical study on the evolution of design patterns. In *ESEC-FSE '07 : Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, pages 385–394, New York, NY, USA, 2007. ACM.

[ADSM05] O. Andriyevska, N. Dragan, B. Simoes, and J. I. Maletic. Evaluating uml class diagram layout based on architectural importance. In *VISSOFT '05 : Proceedings of the 3rd IEEE International Workshop on Visualizing Software for Understanding and Analysis*, page 9, Washington, DC, USA, 2005. IEEE Computer Society.

[AIS78] Christopher Alexander, Sara Ishikawa, and Murray Silverstein. *A Pattern Language : Towns, Buildings, Construction (Center for Environmental Structure Series)*. Oxford University Press, New York, August 1978.

[AK08] Andrey Andreev and Nikolay Kirov. *Text search in document images based on Hausdorff distance measures*. New York, NY, USA, 2008.

[AM72] M. Appelmans and J. Michiels. *Leçons sur les maladies des yeux*. Éditions Peeters, Louvain, 1972.

[BEM+96] Lionel Briand, Khaled El, Emam Sandro Morasca, Centre De Recherche Informatique De, Centre De Recherche Informatique De, and Politecnico Di Milano. On the application of measurement theory in software engineering. *Journal of Empirical Software Engineering*, 1 :61–88, 1996.

[BPF03] S. Jason Babcock, Jeff B. Pelz, and Mark D. Fairchild. Eye tracking observers during rank order, paired comparison, and graphical rating tasks. In *The PICS Conference, International Technical Conference on The Science and Systems of Digital Photography*, pages 10–15, Rochester, NY, May 2003.

[Bro78] Ruven Brooks. *Using a behavioral theory of program comprehension in software engineering*. IEEE Press, Piscataway, NJ, USA, 1978.

[BT04] Roman Bednarik and Markku Tukiainen. *Visual attention tracking during program debugging*, pages 331–334. ACM, New York, NY, USA, 2004.

[BT06] Roman Bednarik and Markku Tukiainen. An eye-tracking methodology for characterizing program comprehension processes. pages 125–132, 2006.

[BW08] Sarah Boslaugh and Paul Watters. *Statistics In A Nutshell*. O'Reilly, 2008.

[CDEDSAT05] Centre Canadien D'Hygiène Et De Sécurité Au Travail. Fatigue, 2005.

[CK05] Christopher F. Chabris and Stephen M. Kosslyn. Representational correspondence as a basic principle of diagram design. pages 36–57, 2005.

[Cli96] Marshall P. Cline. The pros and cons of adopting and applying design patterns in the real world, 1996.

[C.S02] T.Cormen C.Leiserson R.Rivest C.Stein. *Introduction à l'algorithmique*. 2002.

[DJ94] M.-P. Dubuisson and A.K. Jain. *A modified Hausdorff distance for object matching*, volume 1. Oct 1994.

[DPCGA08] M. Di Penta, L. Cerulo, Y. G. Gueheneuc, and G. Antoniol. *An empirical study of the relationships between design pattern roles and class change proneness.* 2008.

[Duc07] A. T. Duchowski. Eye tracking methodology : Theory and practice, 2007.

[Dwy01] Tim Dwyer. Three dimensional uml using force directed layout. In *APVis '01 : Proceedings of the 2001 Asia-Pacific symposium on Information visualisation*, pages 77–85, Darlinghurst, Australia, Australia, 2001. Australian Computer Society, Inc.

[DYZ07] Jing Dong, Sheng Yang, and Kang Zhang. *Visualizing Design Patterns in Their Applications and Compositions*, 2007.

[Eic02] Holger Eichelberger. *Aesthetics of Class Diagrams*, page 23. IEEE Computer Society, Washington, DC, USA, 2002.

[Eic03] Holger Eichelberger. *Nice class diagrams admit good design ?*, pages 159–ff. ACM, New York, NY, USA, 2003.

[Eic08] Holger Eichelberger. Automatic layout of uml use case diagrams. In *SoftVis '08 : Proceedings of the 4th ACM symposium on Software visualization*, pages 105–114, New York, NY, USA, 2008. ACM.

[EYG97] A. H. Eden, A. Yehudai, and J. Gil. Precise specification and automatic application of design patterns. In *ASE '97 : Proceedings of the 12th international conference on Automated software engineering (formerly : KBSE)*, page 143, Washington, DC, USA, 1997. IEEE Computer Society.

[FKGS04] Robert B. France, Dae-Kyoo Kim, Sudipto Ghosh, and Eunjee Song. A uml-based pattern specification technique. volume 30, pages 193–206, Piscataway, NJ, USA, 2004. IEEE Press.

[Gam98] Erich Gamma. Applying design patterns in java. pages 105–114, New York, NY, USA, 1998. Cambridge University Press.

[GB98] Normand Grégoire and Mikael Bouillot. Hausdorff distance between convex polygons. http ://cgm.cs.mcgill.ca/ godfried/teaching/cg-projects/98/normand/main.html, 1998.

[GHJV95] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns : Elements of Reusable Object-Oriented Software.* Toronto, Ontario. Canada, 1995.

[GJK+03] Carsten Gutwenger, Michael Jünger, Karsten Klein, Joachim Kupke, Sebastian Leipert, and Petra Mutzel. *A new approach for visualizing UML class diagrams.* ACM, New York, NY, USA, 2003.

[GK99] Joseph H. Goldberg and Xerxes P. Kotval. Computer interface evaluation using eye movements : methods and constructs. Technical Report 6, October 1999.

[GSL+02] Joseph H. Goldberg, Mark J. Stimson, Marion Lewenstein, Neil Scott, and Anna M. Wichansky. Eye tracking in web search tasks : design implications. pages 51–58, 2002.

[Gué06] Yann-Gaël Guéhéneuc. TAUPE : towards understanding program comprehension. pages 1–13, 2006.

[Gué05] Yann-Gaël Guéhéneuc. A theory of program comprehension – Joining vision science and program comprehension. Technical report, University of Montreal, December 2005.

[HALS08] Naji Habra, Alain Abran, Miguel Lopez, and Asma Sellami. A framework for the design and verification of software measurement methods. *J. Syst. Softw.*, 81(5) :633–648, 2008.

[HE03] J. Wolff von Gudenberg H. Eichelberger. Uml class diagrams - state of the art in layout techniques. In *International Workshop on Visualizing Software for Understanding and Analysis*, pages pp. 30–34, Amsterdam, 2003.

[HMM00] Ivan Herman, Guy Melançon, and M. Scott Marshall. *Graph Visualization and Navigation in Information Visualization : A Survey.* PhD thesis, Piscataway, NJ, USA, 2000.

[IEE90] IEEE. Ieee standard glossary of software engineering terminology. *IEEE Std 610.12-1990*, pages –, Dec 1990.

[Jea08] Sébastien Jeanmart. Evaluation de l'impact d'un patron de conception sur la compréhension et la maintenance de programmes - une expérimentation par un système d'eye-tracking. Master's thesis, Facultés Universitaires Notre-Dame de la Paix, 2008.

[JH06] Sune Alstrup Johansen and John Paulin Hansen. Do we need eye trackers to tell where people look ? In *CHI '06 : CHI '06 extended abstracts on Human factors in computing systems*, pages 923–928, New York, NY, USA, 2006. ACM.

[JKD02] Robert J. K. Jacob, Keith S. Karn, and Ph. D. Commentary on section 4. eye tracking in human-computer interaction and usability research : Ready to deliver the promises., August 31 2002.

[JKF01] Oliver Jesorsky, Klaus J. Kirchberg, and Robert Frischholz. *Robust Face Detection Using the Hausdorff Distance*, pages 90–95. Springer-Verlag, London, UK, 2001.

[KB09] David Saff Kent Beck, Erich Gamma. Junit a cook's tour. http ://junit.sourceforge.net/doc/cookstour/cookstour.htm, 2009.

[KG08] Foutse Khomh and Yann-Gaël Guéhéneuc. *Do Design Patterns Impact Software Quality Positively ?*, pages 274–278. IEEE, 2008.

[KKB+98] R. Kazman, M. Klein, M. Barbacci, T. Longstaff, H. Lipson, and J. Carriere. The architecture tradeoff analysis method. In *Engineering of Complex Computer Systems, 1998. ICECCS '98. Proceedings. Fourth IEEE International Conference on*, pages 68–78, 1998.

[KNW99] Oliver Karch, Hartmut Noltemeier, and Thomas Wahl. *Robot Localization Using Polygon Distances*, pages 200–219. Springer-Verlag, London, UK, 1999.

[KR90] L. Kaufman and P. Rousseeuw. *Finding Groups in Data. An Introduction to Cluster Analysis.* Wiley-Interscience, New York, 1990.

[Lau07] Louis Laurencelle. Inventer ou estimer la puissance statistique ? quelques considérations utiles pour le chercheur. *Tutorials in Quantitative Methods for Psychology*, 3(2) :35–42, 2007.

[LK98] Anthony Lauder and Stuart Kent. Precise visual specification of design patterns. In *ECCOP '98 : Proceedings of the 12th European Conference on Object-Oriented Programming*, pages 114–134, London, UK, 1998. Springer-Verlag.

[MB01] William B. Mcnatt and James M. Bieman. Coupling of design patterns : Common practices and their benefits. In *Proc. COMPSAC 2001. To Appear*, pages 574–579. IEEE Computer Society Press, 2001.

[MHG02] David Mapelsden, John Hosking, and John Grundy. Design pattern modelling and instantiation using dpml. Darlinghurst, Australia, Australia, 2002.

[NdS01] Institut Nationale de Statistique. Enquête 2001 socio-économique générale : Dossier de presse. Septembre 2001.

[Nic94] Jeffrey Nickerson. *Visual Programming.* Computer science, New York University, New York, NY, USA, 1994.

[Pal99] S. E. Palmer. *Vision Science-Photons to Phenomenology.* MIT Press, Cambridge, MA, 1999.

[PB04] Alex Poole and J. Linden Ball. *Eye Tracking in Human-Computer Interaction and Usability Research : Current Status and Future Prospects.* In Ghaoui, Claude (Ed.), 2004.

[PBG+08] Sami Pietinen, Roman Bednarik, Tatiana Glotova, Vesa Tenhunen, and Markku Tukiainen. *A method to study visual attention aspects of collaboration : eye-tracking pair programmers simultaneously*, pages 39–42. ACM, New York, NY, USA, 2008.

[PCM+01] Helen C. Purchase, Linda Colpoys, Matthew McGill, David Carrington, and Carol Britton. *UML class diagram syntax : an empirical study of comprehension.* Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 2001.

[PLL08] Bo Gun Park, Kyoung Mu Lee, and Sang Uk Lee. *Color-based image retrieval using perceptually modified Hausdorff distance.* New York, NY, United States, 2008.

[PMCC01] Helen C. Purchase, Matthew McGill, Linda Colpoys, and David Carrington. *Graph drawing aesthetics and the comprehension of UML class diagrams : an empirical study.* PhD thesis, Darlinghurst, Australia, Australia, 2001.

[Por08] Gerardo Cepeda Porras. Analyse à l'aide d'occulomètres, de techniques de visualisation uml de patrons de conception pour la compréhension de programmes. Technical report, 2008.

[PPV00] Dewayne E. Perry, Adam A. Porter, and Lawrence G. Votta. Empirical studies of software engineering : a roadmap. In *ICSE '00 : Proceedings of the Conference on The Future of Software Engineering*, pages 345–355, New York, NY, USA, 2000. ACM.

[PULPT02] Lutz Prechelt, Barbara Unger-Lamprecht, Michael Philippsen, and Walter F. Tichy. *Two Controlled Experiments Assessing the Usefulness of Design Pattern Documentation in Program Maintenance.* Piscataway, NJ, USA, 2002.

[Pur97] Helen C. Purchase. *Which Aesthetic has the Greatest Effect on Human Understanding ?* Springer-Verlag, London, UK, 1997.

[PUT+01] L. Prechelt, B. Unger, W. F. Tichy, P. Brössler, and L. G. Votta. *A Controlled Experiment in Maintenance Comparing Design Patterns to Simpler Solutions.* Piscataway, NJ, USA, 2001.

[Ray98] K. Rayner. Eye movements in reading and information processing : 20 years of research. *Psychol Bull*, 124(3) :372–422, November 1998.

[RJB04] James Rumbaugh, Ivar Jacobson, and Grady Booch. *Unified Modeling Language Reference Manual, The (2nd Edition) (Addison-Wesley Object Technology Series).* Addison-Wesley Professional, July 2004.

[Rot91] Günter Rote. *Computing the minimum Hausdorff distance between two point sets on a line under translation*, volume 38, pages 123–127. Elsevier North-Holland, Inc., Amsterdam, The Netherlands, The Netherlands, 1991.

[RP07] Deborah Rumsey PhD. *Intermediate Statistics For Dummies.* For Dummies, 2007.

[RT06] Aude Richter and Michèle Ternaux. Les mouvements oculaires permettent la saisie des images et assurent leur stabilité sur la rétine. http ://acces.inrp.fr/acces/ressources/neurosciences/vision, 2005-2006.

[SG00] Dario D. Salvucci and Joseph H. Goldberg. *Identifying fixations and saccades in eye-tracking protocols.* New York, NY, USA, 2000.

[Sjo07] *The Future of Empirical Methods in Software Engineering Research*, May 2007.

[SK98] R. Schauer and R. Keller. Pattern visualization for software comprehension. page 4, 1998.

[SR 06] SR Research Ltd., Mississauga, Canada. *EyeLink II User Manual*, 2006.

[SSC08] Frederick Shic, Brian Scassellati, and Katarzyna Chawarska. The incomplete fixation measure. In *ETRA '08 : Proceedings of the 2008 symposium on Eye tracking research & applications*, pages 111–114. ACM, New York, NY, USA, 2008.

[SW05] Dabo Sun and Kenny Wong. On evaluating the layout of uml class diagrams for program comprehension. In *IWPC '05 : Proceedings of the 13th International Workshop on Program Comprehension*, pages 317–326, Washington, DC, USA, 2005. IEEE Computer Society.

[TH03] Scott Tilley and Shihong Huang. A qualitative assessment of the efficacy of uml diagrams as a form of graphical documentation in aiding program understanding. In *SIGDOC '03 : Proceedings of the 21st annual international conference on Documentation*, pages 184–191, New York, NY, USA, 2003. ACM.

[TT07] Tim Trese and Scott Tilley. *Documenting software systems with views V : towards visual documentation of design patterns as an aid to program understanding.* ACM, New York, NY, USA, 2007.

[Ven05] Bill Venners. How to use design patterns : A conversation with erich gamma, part 1, May 2005.

[Vli98] John Vlissides. Pattern hatching notation, notation, notation c++ report, April 1998.

[vMV95] Anneliese von Mayrhauser and A. Marie Vans. *Program Comprehension During Software Maintenance and Evolution*, volume 28, pages 44–55. IEEE Computer Society Press, Los Alamitos, CA, USA, 1995.

[Wen01] Peter Wendorff. Assessment of design patterns during software reengineering : Lessons learned from a large commercial project. In *CSMR '01 : Proceedings of the Fifth European Conference on Software Maintenance and Reengineering*, page 77, Washington, DC, USA, 2001. IEEE Computer Society.

[Wik09] Wikipédia. Distance de hausdorff. `http://fr.wikipedia.org/wiki/Distance_de_Hausdorff`, Août 2009.

[WPCM02] Colin Ware, Helen Purchase, Linda Colpoys, and Matthew McGill. Cognitive measurements of graph aesthetics. In *Information Visualization*, volume 1, pages 103–110. Palgrave Macmillan, 2002.

[WRH+00] Claes Wohlin, Per Runeson, Martin Hösrt, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. *Experimentation in Software Engineering*, 2000.

[YKM07] Shehnaaz Yusuf, Huzefa Kagdi, and Jonathan I. Maletic. Assessing the comprehension of uml class diagrams via eye tracking. *ICPC '07 : Proceedings of the 15th IEEE International Conference on Program Comprehension*, pages 113–122, 2007.

Annexe A

Tableaux statistiques

Remarques :
- les résultats (sauf les données brutes) présentés ici ont été effectués après la réduction de données présentée à la section 8.2 ;
- les résultats statistiquement significatifs (inférieurs à 0.05) ont été encadrés. La signification de ceux-ci varie en fonction du test réalisé et est donc rappelée en début de sous-section ;
- les calculs inutiles n'ont pas été effectués et sont donc absents. Ils sont remplacé par '-' ;
- les calculs ont été réalisés avec le langage R [1].

A.1 Données brutes

Données relatives aux sujets

Sujet	Cluster	Statut	Experience	Hreveil	Hpassage	NuitNormale	NuitActuelle
Pixys01	2	pro	5	6,5	14	6,5	6
Pixys02	2	pro	4	6,5	15	8	7,5
Pixys03	2	pro	4	6,5	10	7	6
Pixys04	2	pro	4	5	11	6,5	7
Pixys05	1	pro	4	7,3	13	8	8,5
Pixys06	2	pro	3	7,75	14	7	4
Pixys07	1	pro	3	8	10	9	9
Pixys08	2	pro	2	7,5	11,125	8,5	5
Sujet01	2	stud	2	8	15	8	7
Sujet03	2	stud	1	8	14	8	8
Sujet05	1	stud	1	7,5	17	6,5	9
Sujet06	0	stud	2	8,5	13	8	8,5
Sujet07	2	stud	4	7	13	6	8
Sujet08	2	stud	2	12	14	5	5,5
Sujet09	2	stud	5	10	13	6	6
Sujet10	2	stud	2	9	15	5,5	6
Sujet11	1	stud	1	9	14	7,5	6
Sujet12	2	stud	2	8	15,125	7	8
Sujet13	1	stud	5	6	15	6	5
Sujet14	2	stud	5	9	11	9	8
Sujet15	2	pro	3	6	18	8	6

[1] `http://www.r-project.org/`

Données relatives aux tests des sujets

TF : Total de fixations
TFM : Temps de fixation moyen
RFO : Rapport de fixation on targe
DS : Densité spatiale
MT : Matrice de transition
RJNS : Réponse juste non sémantiquement
RJS : Réponse juste sémantiquement
RF : Réponse fausse
REP : Réponse
Comp : Composite
Obs : Observateur
Sans : Sans patron

SUJET	TF	TFM	RFO	DS	MT	RJNS	RJS	RF	REP	PATRON
Pixys01	422	327,1943	0,3342	0,3094	0,0025	0	1	0	1	Comp
Pixys02	1824	268,3399	0,4062	0,6469	0,0113	0	0	1	0	Comp
Pixys03	1318	325,6176	0,3363	0,5531	0,0077	0	0	1	0	Comp
Pixys04	636	314,956	0,186	0,4719	0,0039	0	0	1	0	Comp
Pixys05	251	308,3665	0,567	0,2969	0,0019	1	0	0	0,5	Comp
Pixys06	751	259,1691	0,562	0,4344	0,0042	1	0	0	0,5	Comp
Pixys07	669	340,3767	0,8085	0,3594	0,004	1	0	0	0,5	Comp
Pixys08	424	420,4906	0,5665	0,3438	0,0029	1	0	0	0,5	Comp
Sujet01	533	232,2176	0,46	0,4219	0,0038	0	0	1	0	Comp
Sujet03	274	289,4015	0,2602	0,2281	0,0018	0	0	1	0	Comp
Sujet05	833	304,0912	0,7806	0,3813	0,0048	0	0	1	0	Comp
Sujet06	195	591,5282	0,0741	0,1875	0,0012	0	0	1	0	Comp
Sujet07	644	248,4348	0,8013	0,3875	0,0042	0	1	0	1	Comp
Sujet08	514	306,9416	0,6127	0,3656	0,0029	0	0	1	0	Comp
Sujet09	284	245,5915	0,6303	0,2719	0,0021	0	1	0	1	Comp
Sujet10	604	291,6556	0,4236	0,2531	0,0026	0	0	1	0	Comp
Sujet11	1057	242,1722	0,6592	0,55	0,0064	0	0	1	0	Comp
Sujet12	422	263,9716	0,6667	0,2844	0,0025	0	0	1	0	Comp
Sujet13	1231	364,5524	0,739	0,4656	0,0051	1	0	0	0,5	Comp
Sujet14	588	227,7551	0,6234	0,4938	0,0042	1	0	0	0,5	Comp
Sujet15	1539	331,0019	0,4677	0,5719	0,0078	1	0	0	0,5	Comp
Pixys01	1058	300,4423	0,8551	0,5031	0,0059	1	0	0	0,5	Sans
Pixys02	982	252,1548	0,7016	0,5813	0,0069	1	0	0	0,5	Sans
Pixys03	646	317,1641	0,71	0,4094	0,0039	1	0	0	0,5	Sans
Pixys04	390	380,1026	0,8204	0,4188	0,0028	1	0	0	0,5	Sans
Pixys05	262	299,145	0,831	0,2313	0,0016	1	0	0	0,5	Sans
Pixys06	1013	264,6752	0,792	0,6	0,0067	1	0	0	0,5	Sans
Pixys07	317	289,6151	0,875	0,2406	0,0022	1	0	0	0,5	Sans
Pixys08	299	415,291	0,4785	0,3031	0,0023	0	0	1	0	Sans
Sujet01	451	224,6563	0,7679	0,4094	0,0033	0	0	1	0	Sans
Sujet03	181	313,3481	0,7436	0,1781	0,0012	0	0	1	0	Sans
Sujet05	396	310,9495	0,9494	0,2594	0,0025	1	0	0	0,5	Sans
Sujet06	84	437,2857	1	0,0844	5,96E-004	1	0	0	0,5	Sans
Sujet07	669	265,0703	0,6909	0,4563	0,0045	1	0	0	0,5	Sans
Sujet08	338	349,4793	0,8217	0,2656	0,0022	1	0	0	0,5	Sans
Sujet09	524	217,1298	0,8598	0,3844	0,0034	0	1	0	1	Sans
Sujet10	280	275,6857	0,7411	0,2094	0,0018	1	0	0	0,5	Sans
Sujet11	964	287,8838	0,6956	0,5125	0,0064	1	0	0	0,5	Sans
Sujet12	444	256,045	0,8667	0,3031	0,0027	1	0	0	0,5	Sans
Sujet13	598	312,1472	0,924	0,3844	0,004	1	0	0	0,5	Sans
Sujet14	769	173,3316	0,7449	0,5406	0,0055	0	1	0	1	Sans
Sujet15	1512	346,254	0,7006	0,6031	0,0086	1	0	0	0,5	Sans
Pixys01	889	328,7154	0,306	0,3656	0,004	0	1	0	1	Obs
Pixys02	457	272,9803	0,5031	0,4219	0,0033	0	1	0	1	Obs
Pixys03	1564	310,1407	0,6384	0,5781	0,0084	0	1	0	1	Obs
Pixys04	665	344,6256	0,5124	0,4313	0,0045	0	1	0	1	Obs
Pixys05	134	276,3284	0,9633	0,1188	8,30E-004	1	0	0	0,5	Obs
Pixys06	1217	262,235	0,7171	0,6188	0,0062	1	0	0	0,5	Obs

SUJET	TF	TFM	RFO	DS	MT	RJNS	RJS	RF	REP	PATRON
Pixys07	614	294,7036	0,6296	0,4063	0,0044	0	1	0	1	Obs
Pixys08	430	391,1349	0,7403	0,3438	0,0029	0	1	0	1	Obs
Sujet01	243	221,3663	0,6284	0,2563	0,0017	0	0	1	0	Obs
Sujet03	868	272,2258	0,5572	0,4281	0,0045	0	0	1	0	Obs
Sujet05	684	313,7895	0,75	0,3625	0,0036	0	0	1	0	Obs
Sujet06	373	409,63	0,2515	0,2813	0,0024	1	0	0	0,5	Obs
Sujet07	552	255,4638	0,6012	0,3344	0,003	0	1	0	1	Obs
Sujet08	283	339,1519	0,75	0,2031	0,0016	1	0	0	0,5	Obs
Sujet09	357	265,8599	0,6128	0,3344	0,0026	0	1	0	1	Obs
Sujet10	737	287,4953	0,7817	0,4125	0,0041	0	0	1	0	Obs
Sujet11	1158	251,7686	0,5354	0,5625	0,0072	1	0	0	0,5	Obs
Sujet12	762	254,4619	0,6695	0,4156	0,0045	0	1	0	1	Obs
Sujet13	776	300,0052	0,5624	0,4781	0,0049	1	0	0	0,5	Obs
Sujet14	557	193,1921	0,7038	0,3969	0,0037	1	0	0	0,5	Obs
Sujet15	1386	367,0794	0,8134	0,4906	0,0062	0	1	0	1	Obs

A.2 Résultats des tests de normalité

Le test utilisé est celui de Kolmogorov-Smirnov. Une valeur significative au test signifie que la population de l'échantillon n'est pas normale. Les tests d'hypothèse paramétrique ne peuvent donc pas tous s'y appliquer. Dans ce cas de figure, une transformation logarithmique de données a été effectuée afin de tenter d'obtenir une population normale.

La première sous-section donne les résultats pour les données brutes tandis que la seconde donne les résultats après la transformation logarithmique si celle-ci nous permet d'obtenir une population normale.

A.2.1 Tests de normalité sur les données brutes

Données groupées suivant les clusters

Groupe	Mesure	Sans patron	Observateur	Objet composite
Groupe 1	RFO	0.8518	0.5906	0.7917
	DS	0.3276	0.4533	0.9538
	MT	0.1525	0.6151	0.6969
	TFM	0.5377	0.6692	0.5354
Groupe 2	RFO	0.0616	0.8838	0.8727
	DS	0.7694	0.4879	0.9640
	MT	0.5744	0.1038	0.0019
	TFM	0.9997	0.8886	0.4025

Données groupées suivant la quantité de sommeil

Groupe	Mesure	Sans patron	Observateur	Objet composite
Assez dormi (AD)	RFO	0.2922	0.6969	0.4053
	DS	0.1773	0.5208	0.5689
	MT	0.5490	0.2348	0.4668
	TFM	0.7318	0.5929	0.5290
Pas assez dormi (PAD)	RFO	0.0493 [2]	0.8576	0.8642
	DS	0.5583	0.8858	0.7608
	MT	0.6870	0.6944	0.3452
	TFM	0.8452	0.9865	0.3976

Données groupées suivant le statut

Groupe	Mesure	Sans patron	Observateur	Objet composite
Professionnel	RFO	0.2498	0.8324	0.5820
	DS	0.8769	0.4046	0.4034
	MT	0.3269	0.3214	0.0126
	TFM	0.2248	0.8119	0.3088
Etudiant	RFO	0.2011	0.6226	0.1364
	DS	0.8769	0.8784	0.6219
	MT	0.4837	0.8730	0.6625
	TFM	0.6463	0.7817	0.3583

A.2.2 Tests de normalité sur les données transformées via le logarithme

Données groupées suivant les clusters

Groupe	Mesure	Sans patron	Objet composite
Groupe 1	MT	0.4557	0.2029
Groupe 2	MT	0.9860	0.1650

Données groupées suivant le statut

Groupe	Mesure	Sans patron	Observateur	Objet composite
Professionnel	MT	0.5024	0.1842	0.2313
Etudiant	MT	1	0.5299	0.9090

A.3 Résultats des tests d'homoscédasticité

Le test utilisé est celui de Breusch-Pagan. Une valeur significative au test signifie que les populations des échantillons n'ont pas les variances homogènes. Le test d'Anova ne peut dès lors pas s'appliquer.

Les colonnes intitulées « Observateur » et « Composite » sont à mettre en relation avec le tableau nommé « 2-way Anova : patrons de conception et clusters » dans la section A.5. La colonne intitulée « Sommeil » est à mettre en relation avec le tableau nommé « 2-way Anova : patrons de

[2] Il s'agit de la seule population pour laquelle la transformation logarithmique ne donne pas une population normale.

conception et groupes définis suivant la quantité de sommeil » et la colonne intitulée « Statut » avec le tableau nommé « 2-way Anova : patrons de conception et groupes définis suivant le statut » de cette même section.

Facteur	Observateur	Composite	Sommeil	Statut
RFO	0.5663	0.1017	-	0.1633
DS	0.4539	0.5763	0.7887	0.6792
MT	0.9342	0.7632 [3]	0.0566	0.7857 [3]
TFM	0.0534	0.1594	0.1661	0.9290

A.4 Moyennes des échantillons

Les données présentées dans cette sous-section ne constituent pas des résultats de test. Il n'y a donc pas de valeur statistiquement significative.

Données groupées suivant les clusters

Groupe	Mesure	Sans patron	Observateur	Objet composite
Groupe 1	RFO	0.8477	0.6896	0.6833
	DS	0.3616	0.4232	0.3960
	MT	0.0037	0.0045	0.0041
	TFM	288.64	279.04	297.53
	PCA	0.5000	0.5714	0.2857
Groupe 2	RFO	0.7412	0.6268	0.4699
	DS	0.4048	0.3844	0.4091
	MT	0.0040	0.0039	0.0044
	TFM	294.62	296.11	294.58
	PCA	0.4615	0.6923	0.3462

Données groupées suivant la quantité de sommeil

Groupe	Mesure	Sans patron	Observateur	Objet composite
Assez dormi (AD)	RFO	-	-	-
	DS	0.2947	0.3447	0.3300
	MT	0.0025	0.0034	0.0031
	TFM	295.66	290.41	291.38
	PCA	0.5000	0.0600	0.3000
Pas assez dormi (PAD)	RFO	-	-	-
	DS	0.4847	0.4513	0.4790
	MT	0.0053	0.0048	0.0056
	TFM	289.40	289.86	299.85
	PCA	0.4500	0.7000	0.3505

[3] Mesures effectuées après une transformation logarithmique

Données groupées suivant le statut

Groupe	Mesure	Sans patron	Observateur	Objet composite
Professionnel	RFO	0.7516	0.6471	0.4705
	DS	0.4323	0.4195	0.4431
	MT	0.0045	0.0045	0.0051
	TFM	318.32	316.44	321.72
	PCA	0.4444	0.8889	0.3889
Etudiant	RFO	0.8005	0.6502	0.6052
	DS	0.3548	0.3804	0.3730
	MT	0.0034	0.0038	0.0037
	TFM	271.43	268.62	274.25
	PCA	0.5000	0.4450	0.2727

A.5 Résultats des tests Anova

Une valeur statistiquement significative au test d'Anova signifie qu'il est envisageable de réfuter l'hypothèse de base. La ou les hypothèses alternatives semblent donc se confirmer.

2-way Anova : patrons de conception et clusters

Patron analysé	Facteur	RFO	DS	MT [4]	TFM
Observateur	Cluster	0.0394	0.9602	0.8016	0.5195
	Patron de conception	0.0015	0.8423	0.7722	0.8882
	Interaction	0.5848	0.3503	0.4931	0.7563
Objet composite	Cluster	0.0005	0.5174	0.8758	0.9329
	Patron de conception	1.1e-6	0.7200	0.5066	0.8579
	Interaction	0.2092	0.7293	0.8412	0.8048

2-way Anova : patrons de conception et groupes définis suivant la quantité de sommeil

Facteur	RFO	DS	MT	TFM
Patron	-	0.8962	0.7714	0.9470
Sommeil	-	4.7e-7	8.1e-6	0.9675
Interaction	-	0.4279	0.4585	0.9053

2-way Anova : patrons de conception et groupes définis suivant le statut

Facteur	RFO	DS	MT [5]	TFM
Patron	1.8e-5	0.9307	0.7968	0.9326
Statut	0.0988	0.0587	0.1480	0.0002
Interaction	0.3462	0.8754	0.9119	0.9994

[4] Le résultats pour l'« Objet composite » sont calculés après une transformation logarithmique
[5] Mesures effectuées après une transformation logarithmique

Annexe B

Taupe

À la base, Taupe à été réalisé par Yann Gaël Guéhéneuc et signifie : « Thoroughly Analysing the Understanding of Programs through Eyesight ». Son but était de visualiser les données fournies par le système d'oculométrie EyeLink II. Ce programme est développé en Java et utilise une API fournie par la compagnie à l'origine d'EyeLink II.

La succession des étudiants qui y ont apporté des modifications a fait que le code était devenu confus et incohérent. Il y avait des fonctionnalités implémentées plusieurs fois, des parties d'interface graphique inutilisables, etc.

Nous avons décidé de nous inspirer de ce travail et de ré-écrire les parties basiques, les parties dont nous avions besoin et d'ajouter les fonctionnalités absentes utiles à notre expérience. De plus, l'API fournie par SR Research ne fonctionne qu'avec le système d'exploitation Windows. Nous avons dès lors contourné l'utilisation de cette API afin de pouvoir travailler sur un système d'exploitation de type GNU/Linux. Nous n'avons pas développé d'interface graphique poussée et avons privilégié une interface en ligne de commande. L'intérêt de Taupe ne réside pas, dans notre cas, dans la visualisation graphique mais dans le traitement des données.

B.1 Architecture

La section suivante présente les différents packages du logiciel et donne un aperçu des fonctionnalités qu'ils fournissent.

agnes. Ce package fait office de librairie permettant l'application de la technique de classification AGNES (section 4.3.2). Afin de l'utiliser, il suffit de lui fournir une matrice de dissimilarité. L'objet représentant cette matrice n'a qu'à implémenter une interface définie.

hausdorff. Ce package fait office de librairie fournissant les opérations nécessaires au calcul de la distance de Hausdorff (section 4.3.1).

convexHull. Ce package fait office de librairie permettant de calculer l'enveloppe convexe d'un ensemble de points. L'algorithme utilisé est une adaptation orientée-objet de celui proposé dans [C.S02].

data. Ce package renferme l'ensemble des données extraites des tests ainsi que les opérations que l'on peut effectuer dessus. Il comprend aussi les informations relatives aux *stimuli* : zones d'intérêt (pertinentes) et diagrammes (fichiers jpeg).

parsers. Nous avons repris le parser du programme original et lui avons ajouté les fonctionnalités permettant de récupérer l'ensemble des données fournies par les fichiers de données EDF de

EyeLink II. Ce parser ne prenait en compte pour les fixations que les coordonnées et l'instant où elles se produisaient, alors que les informations suivantes étaient également disponibles : œil à l'origine des données, instant de début/fin, coordonnées et taille de la pupille. Bien que nous n'utilisons pas toutes ces informations, les utilisations futures du logiciel pourraient le nécessiter et, l'extraction n'étant pas problématique, nous l'avons ajoutée. Ces mêmes informations sont disponibles pour les saccades.

De plus, nous avons écrit un parser prenant en charge les données sous forme de fichier xml obtenu à l'aide d'un petit utilitaire qui transforme les fichiers EDF d'origine. Ce petit utilitaire étant fonctionnel avec le logiciel Wine sous les systèmes de type GNU/Linux, cela nous permet d'utiliser uniquement un système d'exploitation de ce type.

Une série d'autres parsers sont également écrits afin de récupérer diverses informations : les zones d'intérêt, les zones d'intérêt pertinentes, les décalages, la justesse des réponses et les images des diagrammes.

graphics. Ce package est très petit et n'a pas beaucoup été développé. Il permet de visualiser les données d'un test. Actuellement, une simple commande donnant le numéro du test permet de faire appel à cette fonctionnalité.

metric. Ce package regroupe l'ensemble des mesures que Taupe est capable de fournir concernant les données. Nous y avons implémenté les mesures décrites dans la section 7.2.3.

command. Nous définissons 2 types de commandes :

system. Ce package regroupe un ensemble de commandes utilisables par le système. Par exemple, l'utilisation du package Hausdorff nécessite quelques paramètres ne devant pas être définis par l'utilisateur.

user. Ce package regroupe l'ensemble des commandes utilisables depuis l'interface textuelle. Il suffit de définir une classe héritant de la classe abstraite AUCommand. Le nom de la classe sera alors le nom de la commande à appeler. Des paramètres peuvent être ajoutés.

driver. Ce package permet de « personnaliser » le comportement du logiciel. C'est dans celui-ci que se trouve la fonction « main » où l'utilisateur peut avoir envie de modifier le comportement de base du logiciel (*e.g.*, provoquer automatiquement le parsing des données de correction des tests et appliquer les corrections, supprimer automatiquement les fixations dont les coordonnées se trouvent en dehors du *stimulus*, etc. Il suffit à l'utilisateur de modifier une fonction « Init() ». Idéalement, ce fonctionnement devrait être modifié pour être pris en charge via des paramètres de lancement du programme ou par un fichier de configuration.

Nous définissons également dans ce package une classe regroupant l'ensemble des résultats utiles à notre expérience. Cette classe permet également l'écriture de ces données dans un fichier pour une utilisation externe.

Toutes les fonctionnalités de l'ancien système ne sont pas implémentées mais nous avons tenté de rendre la conception plus générique afin que les futurs ajouts s'intègrent de manière plus cohérente avec les parties existantes du système.

Annexe C

Diagrammes et spécifications

ArgoUML is a UML diagramming application comming with a cognitive support.

Each designer has some goals.
Some critics can be created about a design model.
Some decisions can be proposed about critics and each designer can supports some of them.
There is also a todo list for each designer
Cirtics can lead to new items in a designer's todo list.

FIG. C.1: ArgoUML – Sans patron de conception : Spécification

QuickUML is a design object-oriented software with a highly integrated, core set of UML models. Your entire project is presented through a multi-panel window showing use cases, class models, object models, dictionary and code.

The interface is composed of serveral tools that allows you to interact with your model.

FIG. C.2: QuickUML – Observateur : Spécification

JUnit is a simple, open source framework to write and run repeatable tests.
It is an instance of the xUnit architecture for unit testing frameworks.
JUnit features include:
- Assertions for testing expected results
- Test fixtures for sharing common test data
- Test runners for running tests

FIG. C.3: JUnit – Objet composite : Spécification

We want to add a class named "Consultant" capable of adding some items to the todo list of a designer. How would you do that ?
Be specific about the classes / methods / attibutes.

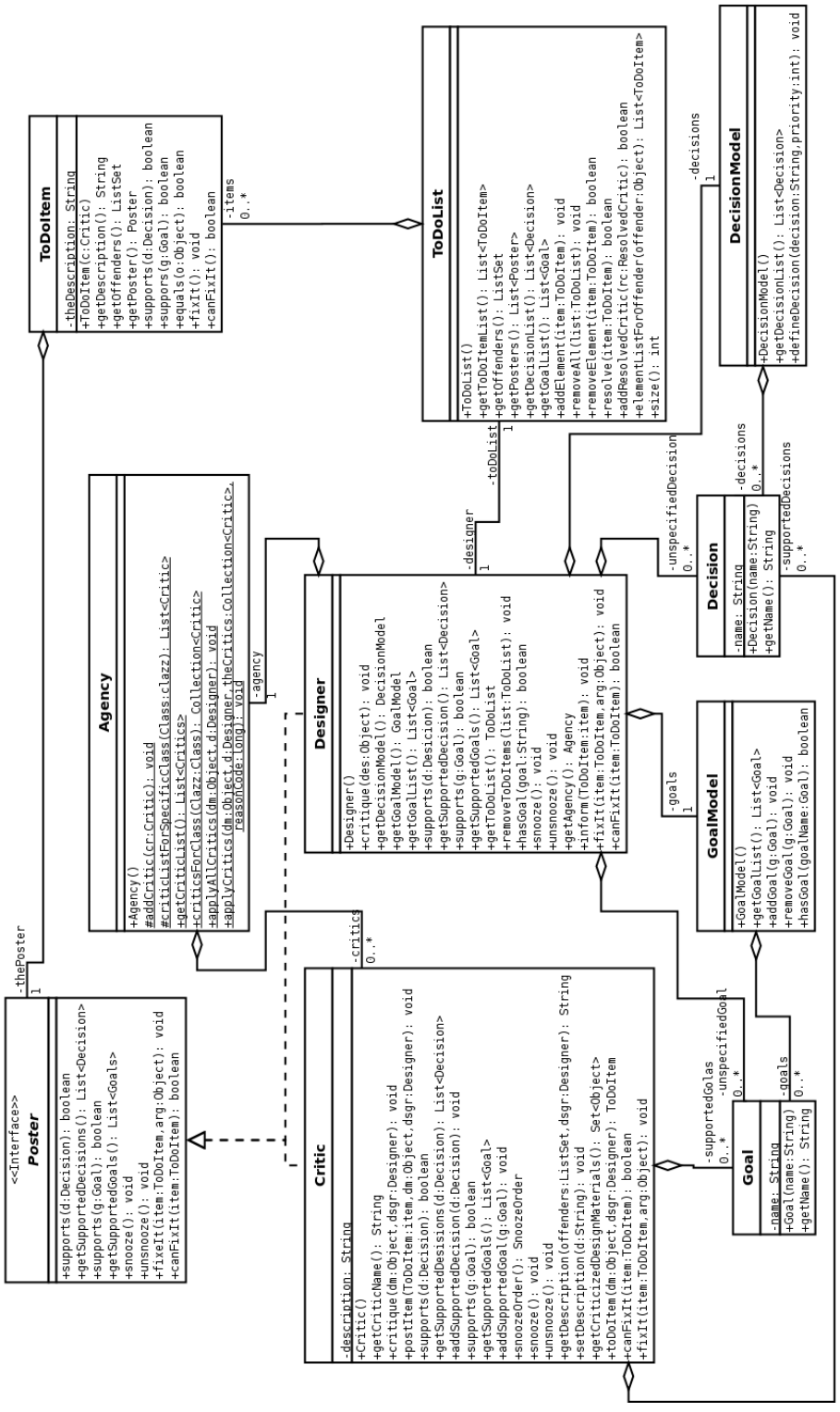


FIG. C.4: ArgoUML – Sans patron de conception : Diagramme

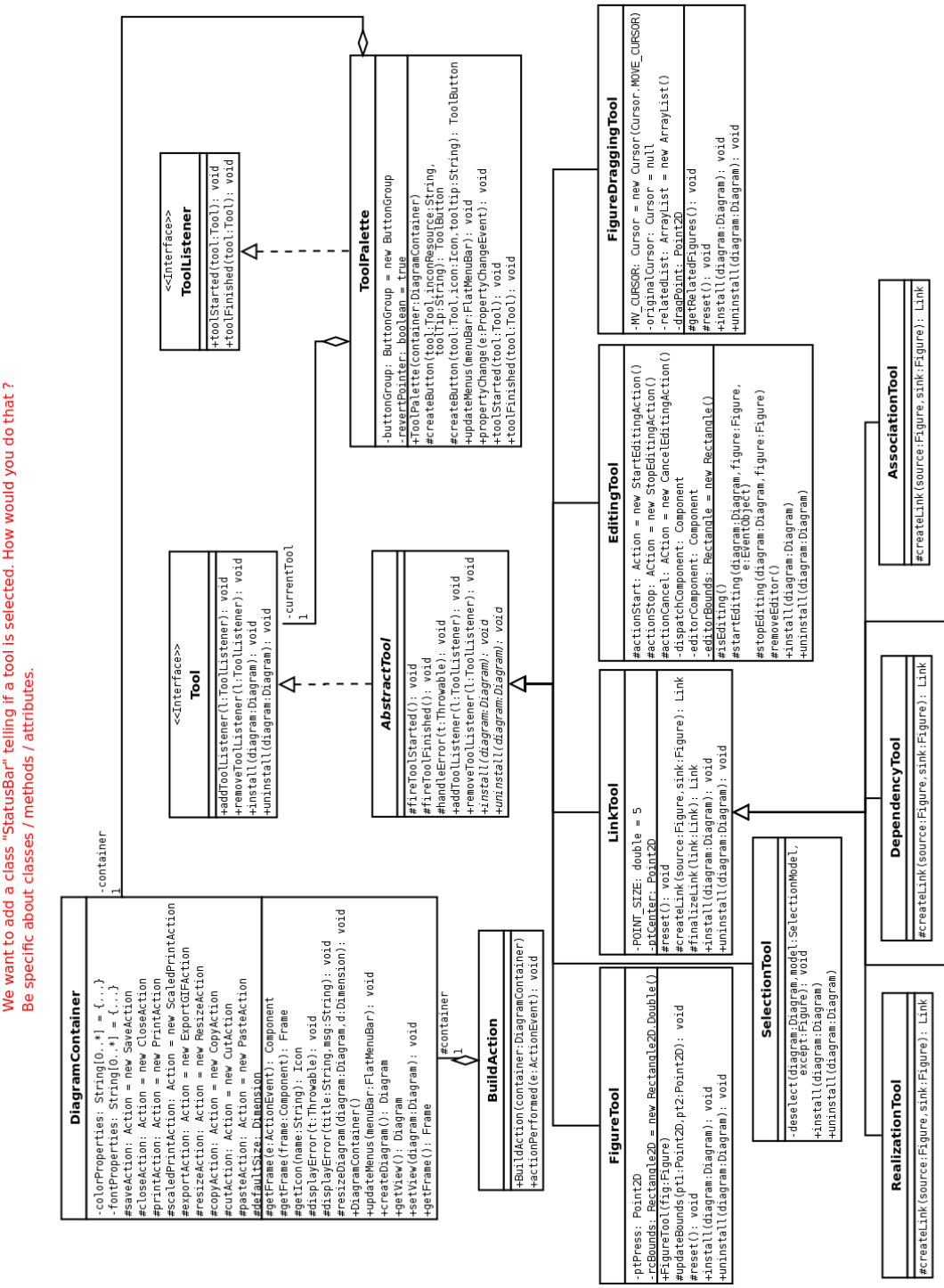


FIG. C.5: QuickUML – Observateur : Diagramme

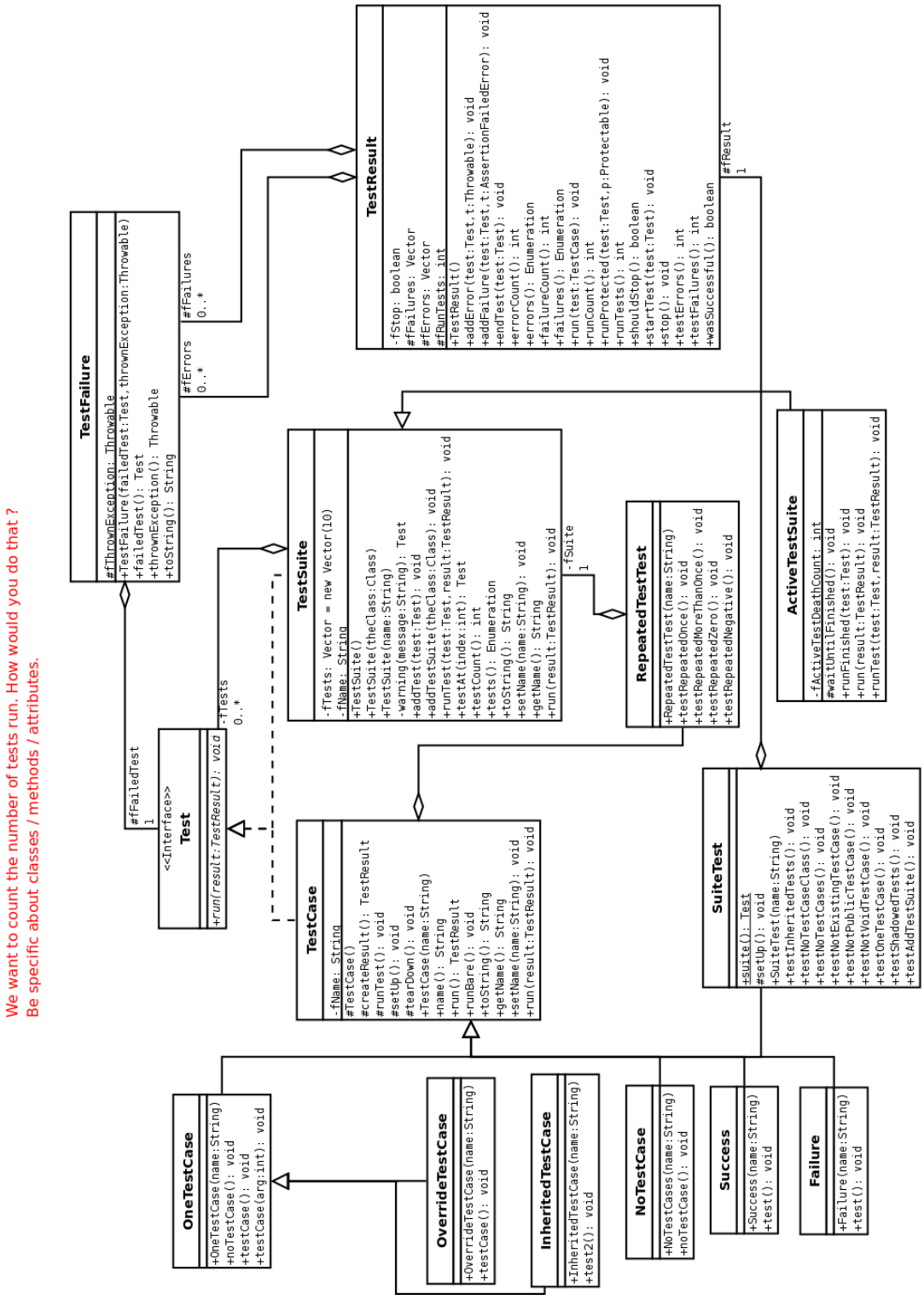


FIG. C.6: JUnit – Objet composite : Diagramme

Annexe D

Checklist de réponses

JUnit

1	Add to inteface «Test» an operation «count»	
2	Define the operation in «TestSuite»	
3	Define the operation in «TestCase»	

1	Tell something about repeatedTestTest	

QuickUML

1	Extends «ToolListener»	
2	Add to «AbstractTool» an operation like «fireToolSelected»	
2'	Add to «AbstractTool» an operation like «fireToolUnselected»	
3	Add to «ToolListener» an operation like «toolSelected»	
3'	Add to «ToolListener» an operation like «toolUnselected»	

ArgoUML

1	Extends «Poster»	
2	Define a method like «postItem»	

or

1	Extends «Critic»	

or

1	Extends «Designer»	
2	Define a method like «postItem»	

Years of experience / Number of project with patterns :_____

Coming hours _____

Questions

How many hours do you need to sleep in general ?

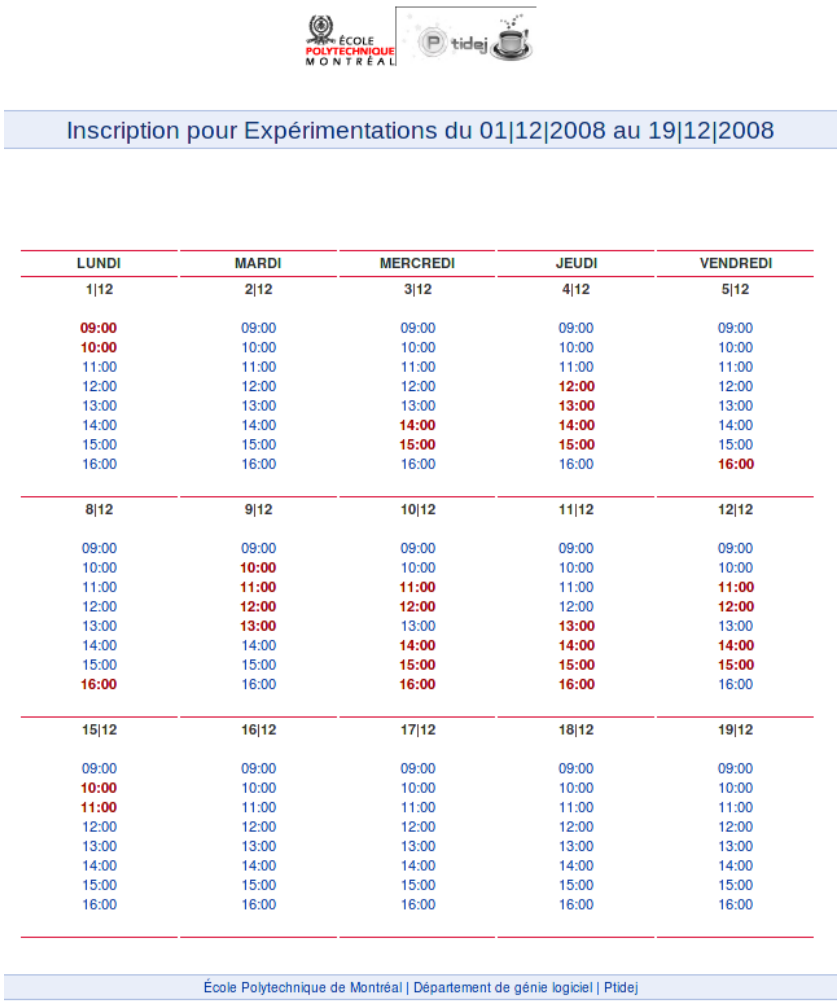
How many hours did you sleep ?

At what time did you wake up ?

Annexe E

Matériel de recrutement

E.1 Site internet d'inscription



Inscription pour Expérimentations du 01|12|2008 au 19|12|2008

LUNDI	MARDI	MERCREDI	JEUDI	VENDREDI
1|12	2|12	3|12	4|12	5|12
09:00	09:00	09:00	09:00	09:00
10:00	10:00	10:00	10:00	10:00
11:00	11:00	11:00	11:00	11:00
12:00	12:00	12:00	**12:00**	12:00
13:00	13:00	13:00	**13:00**	13:00
14:00	14:00	**14:00**	**14:00**	14:00
15:00	15:00	**15:00**	**15:00**	15:00
16:00	16:00	16:00	16:00	**16:00**
8|12	9|12	10|12	11|12	12|12
09:00	09:00	09:00	09:00	09:00
10:00	**10:00**	10:00	10:00	10:00
11:00	**11:00**	**11:00**	11:00	**11:00**
12:00	**12:00**	**12:00**	12:00	**12:00**
13:00	**13:00**	13:00	**13:00**	13:00
14:00	14:00	**14:00**	**14:00**	**14:00**
15:00	15:00	**15:00**	**15:00**	**15:00**
16:00	16:00	**16:00**	**16:00**	16:00
15|12	16|12	17|12	18|12	19|12
09:00	09:00	09:00	09:00	09:00
10:00	10:00	10:00	10:00	10:00
11:00	11:00	11:00	11:00	11:00
12:00	12:00	12:00	12:00	12:00
13:00	13:00	13:00	13:00	13:00
14:00	14:00	14:00	14:00	14:00
15:00	15:00	15:00	15:00	15:00
16:00	16:00	16:00	16:00	16:00

École Polytechnique de Montréal | Département de génie logiciel | Ptidej

FIG. E.1: Page d'accueil du site.

E.2 Affiche

FIG. E.2: Affiche de recrutement

Annexe F

Procédure d'exécution

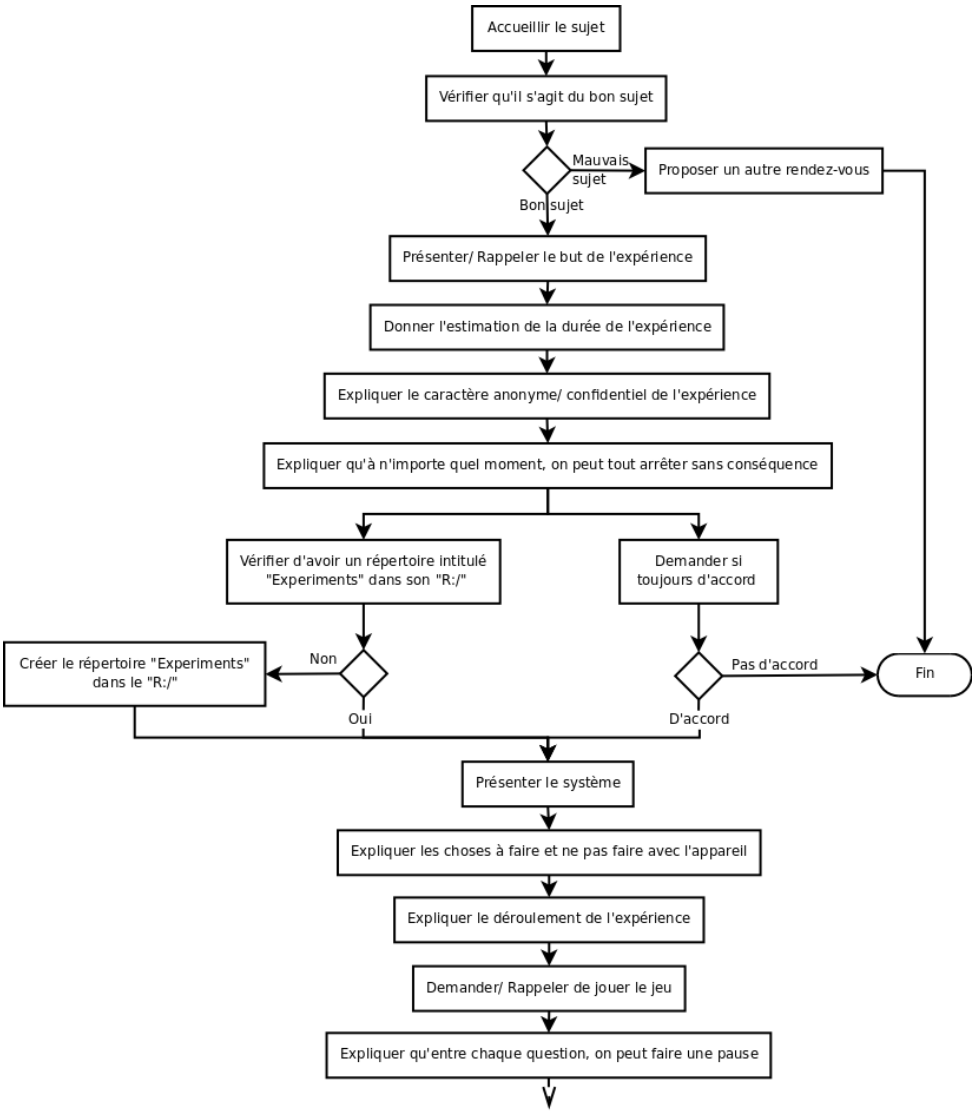


Fig. F.1: Organigramme de l'exécution de l'expérience : partie 1.

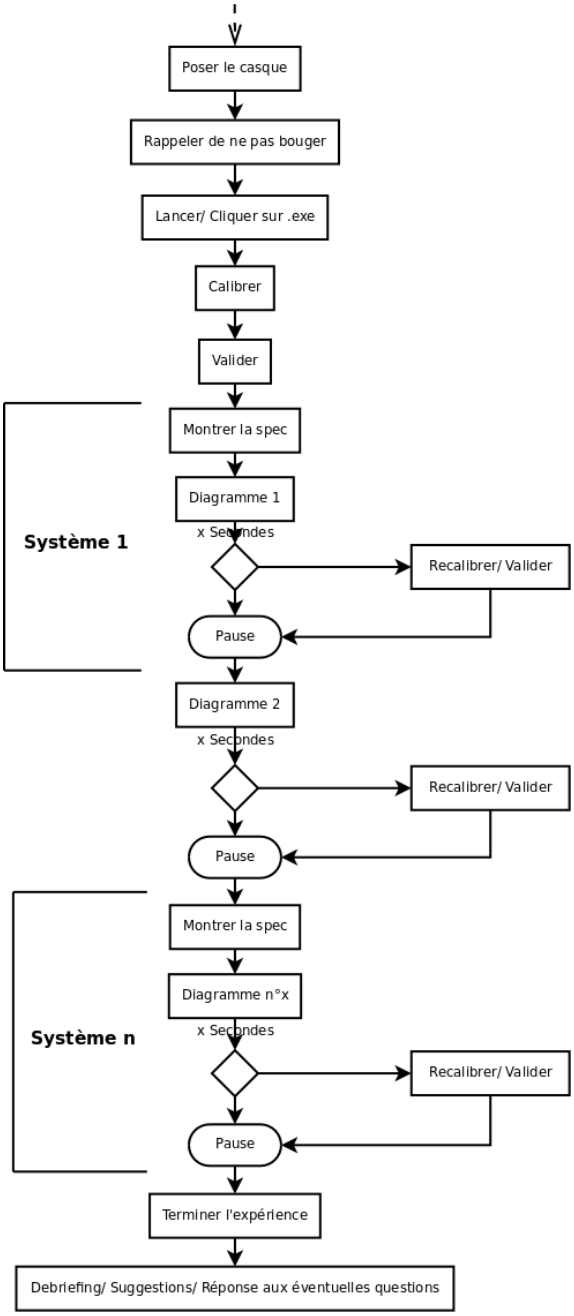


FIG. F.2: Organigramme de l'exécution de l'expérience : partie 2.

Il faut noter que nous n'avons malheureusement pas les sources des diagrammes originaux affichés dans la pièce et qu'il s'agit ici d'une reconstitution.

Annexe G

Rapport de stage

G.1 Introduction

Mon stage a été effectué à l'École Polytechnique de Montréal dans le département de génie logiciel, sous la direction du professeur Yann-Gaël Guéhéneuc et avec le professeur Naji Habra comme promoteur.

Le présent document reprend les informations importantes concernant ce stage. Je commence par une description des objectifs telle qu'elle m'a été faite lors de mon arrivée à Montréal par Mr Guéhéneuc. Je décris ensuite l'environnement de travail dans lequel j'ai évolué. Cette description est suivie d'une section nommée « Méthode de travail » et j'y explique (très brièvement) les différentes tâches relatives à la direction d'une expérience et auxquelles j'ai pris part. Par la suite je donne un aperçu des différentes technologies utilisées durant le stage ainsi que des résultats que j'ai obtenus durant celui-ci. Certaines analyses sont encore en cours à ce jour. Enfin, je termine par une conclusion concernant le stage.

G.2 Objectifs

Les patrons de conception (Design Pattern) décrivent des micro-architectures logicielles, le plus souvent dans le paradigme objets. Ils constituent des solutions largement éprouvées que la littérature recommande d'étudier et utiliser. Cette même littérature leur confère une série d'avantages non négligeables : simplicité, ré-utilisabilité, compréhensibilité, modularité, etc. Mais ces mêmes patrons de conception posent des problèmes pour les chercheurs en génie logiciel dans la mesure où, même si la littérature en vante les bienfaits, elle ne tente pas, ou trop rarement, de les démontrer. Il est très difficile de trouver des recherches tentant d'approuver ou de réfuter la qualité des patrons de conception.

Il n'est pas difficile de soulever des questions concernant ceux-ci :
- Sont-ils vraiment meilleurs que d'autres architectures ?
- Leur connaissance est-elle un pré-requis pour leur utilisation–compréhension ?
- Leur représentation graphique est-elle pertinente ?
- Quels mécanismes entrent en jeu lorsqu'un mainteneur de logiciel les reconnaît ?
- etc.

Le but du stage était de concevoir une expérience permettant d'apporter un nouvel élément de connaissance concernant la compréhension des patrons de conception. Cette expérience devait utiliser un oculomètre. Le choix de l'élément étudié était laissé libre. Mon choix a été de comparer l'impact des patrons de conception sur un groupe de sujets considérés comme « experts » et un groupe de sujets considérés comme « novices ». Le but étant de déterminer si une connaissance des patrons de conception est réellement favorable à leur utilisation et a un impact positif significatif.

Il est à noter que l'équipe dans laquelle je travaillais est l'une des premières au monde à utiliser les systèmes d'oculométrie en génie logiciel expérimental.

G.3 Environnement de travail et compétence de l'équipe

L'équipe dans laquelle je me suis trouvé se nomme « Ptidej » : Pattern Trace Identification, Detection, and Enhancement in Java. Elle se propose d'étudier la qualité sous l'angle de vue des patrons de conception. Comme cela est indiqué sur leur site internet[1], leur but est de développer des théories, des méthodes et des outils pour évaluer et améliorer la qualité des programmes orientés objet par la promotion de l'utilisation d'idiomes, patrons de conception et de patrons architecturaux. Ils tentent de formaliser les patrons de conception, d'identifier leurs occurrences et d'améliorer les occurrences identifiées. Enfin, ils veulent également évaluer expérimentalement l'impact des patrons de conception sur la qualité des programmes orientés objet. C'est sur cette partie de leur recherche que je me suis penché.

L'équipe travaillant sur l'évaluation expérimentale est généralement composée d'étudiants de 2^e cycle et non 3^e cycle. La quantité minimum de travail nécessaire à l'obtention de quelques résultats semble en effet rebuter les candidats du 3^e cycle. Cela est d'ailleurs bien dommage car la conception d'une bonne expérimentation nécessite de l'expérience.

Un point positif de mon stage est que le dernier étudiant à avoir travaillé sur une telle expérience faisait toujours partie de l'équipe, ce qui m'a permis d'être correctement encadré. A deux, nous formions l'équipe travaillant sur la recherche expérimentale.

Un point négatif est que l'équipe est composée d'une part de membres de l'Université de Montréal et d'autre part de membres de l'Ecole Polytechnique. En effet, mon maître de stage étant passé de l'une à l'autre au mois je juin 2008, un bon nombre de l'effectif, du matériel et des compétences se trouvaient encore à l'Université.

G.4 Réalisations

La manière de concevoir une expérimentation n'est pas libre. Elle a été formalisée par Wohlin *et al.* dans [WRH+00]. Wohlin décrit dans ce livre comment écrire et mener une expérience. C'est donc tout naturellement que j'ai suivi le guide proposé par celui-ci.

Le livre propose de mener une expérience selon cinq étapes :

Définition Il s'agit de définir clairement ce sur quoi on veut statuer, ce que l'on veut tester.

Planification Il s'agit de définir concrètement comment on va procéder pour tester nos hypothèses.

Exécution Il s'agit de procéder à l'expérience et à la validation des données récoltées.

Analyse et interprétation Il s'agit d'effectuer les analyses statistiques prévues et les tests d'hypothèse afin de rejeter ou non les hypothèses posées.

Présentation et paquetage Il s'agit ici de reporter le travail effectué afin de permettre à d'autres d'apprendre ou de refaire l'expérience.

Le travail effectué lors du stage peut être résumé en les trois premiers points cités ci-dessus.

Premièrement j'ai dû effectuer un effort de documentation afin de m'imprégner du sujet. J'ai commencé par étudier les travaux de l'équipe relatifs au sujet : [Gué05], [Jea08], [Por08], [Gué06] pour ensuite approfondir ma connaissance avec des ouvrages tels que [Duc07] et [WRH+00] ainsi que divers autres articles. J'ai également effectué des recherches sur les travaux qui ont été conduits à l'aide d'oculomètres. Ce type de travail est en effet fortement conditionné aux types de donnée que le matériel est capable de fournir.

La deuxième étape fut de faire la planification de l'expérience. Il est important de décrire concrètement et précisément chaque étape de l'expérience. Cette partie est très importante et il s'agit d'être rigoureux. Elle prend en compte les études statistiques qui seront menées, une étude des menaces à la validité de l'expérience ainsi qu'une partie d'instrumentation c'est-à-dire la création du matériel nécessaire à l'exécution de l'expérience). Dans mon cas, il s'agissait de sélectionner la manière dont j'allais utiliser les données récoltées en termes statistiques, de créer des diagrammes de classes permettant d'étudier les hypothèses posées, de recruter les sujets pour l'expérience, d'apprendre à utiliser l'oculomètre, etc.

[1] http ://ptidej.dyndns.org/

Enfin, l'exécution de l'expérience a eu lieu à deux endroits différents. J'ai d'abord fait passer des sujets étudiants provenant de l'École Polytechnique de Montréal ainsi que des étudiants provenant de l'Université de Montréal. Ensuite, j'ai déplacé le matériel dans une entreprise nommée Pyxis[2] afin de faire passer l'expérience à des sujets professionnels. Je profite de ces lignes afin de remercier les dirigeants de Pyxis pour m'avoir permis ceci. Il est en effet très rare d'avoir l'opportunité d'avoir des professionnels pour une expérience comme la mienne, le dirigeant n'étant généralement pas très disposé à payer ses employés pour ne pas être productifs.

La suite de l'étude se fait à Namur (en collaboration avec Mr Guéhéneuc) et concerne l'analyse des données récoltées. J'y reviendrai dans la section problèmes rencontrés.

G.5 Technologies utilisées

Selon les phases de l'expérience, différentes technologies ont été utilisées :

- Un outil de reverse engineering (Netbeans) a été utilisé afin de recréer des diagrammes de classe à partir de code source.
- Un logiciel de création de diagrammes (Dia) de classe a été utilisé afin de les faire correspondre aux besoins de l'expérience.
- Il ne m'a pas fallu utiliser d'outil de détection de patrons de conception étant donné que l'équipe possédait déjà une série de logiciels avec la liste des patrons présents dans leur code.
- Un oculomètre, le Eye Link II [3] de l'entreprise SR-Research. Il s'agit d'un système monté sur la tête des sujets permettant d'enregistrer et de suivre en temps réel le mouvement des yeux des sujets. Il est très précis mais impose aux sujets de ne pas bouger la tête. Le système est composé d'un casque et de deux ordinateurs. L'un est destiné au sujet et contient l'ensemble des diagrammes ainsi que le programme qui va piloter l'expérience. L'autre ordinateur permet de piloter le casque, de le calibrer, de voir les mouvements en direct, etc. Il s'agit de l'ordinateur de l'expérimentateur. Le système fonctionne avec des rayons infrarouges qui sont envoyés sur les yeux du sujet afin de surveiller leurs mouvements. Le lancement de l'expérience se fait grâce à une petite application fonctionnant sous Windows qu'il faut programmer en C. Afin de mener une expérience, il faut dès lors écrire l'application qui va agencer les étapes de l'expérience comme vous le voulez.
- Le logiciel pour analyser les données est un logiciel écrit par l'équipe et appelé Taupe : Thoroughly Analysing the Understanding of Programs through Eyesight. Malheureusement, la suite d'étudiants ayant travaillé dessus a rendu le code quelque peu inutilisable. Il comprenait pas mal de redondances et ne fonctionnait que sous Windows. Je l'ai dès lors ré-écrit afin de rendre le code un peu plus modulaire et utilisable pour plusieurs expériences différentes. Auparavant, chaque expérience possédait une fonction « main » lançant les analyses pour cette expérience. Préférant travailler sous Linux, j'ai cherché à le faire tourner dessus. La société SR-Research fournit une librairie dll qu'il faut charger dans le programme Java afin d'extraire les données contenues dans un fichier binaire, ce qui empêche l'utilisation sous Linux. J'ai trouvé un petit logiciel permettant de convertir les fichiers binaires en fichier XML, me permettant dès lors d'extraire les données depuis cet OS.

G.6 Problèmes rencontrés

Le but de mon expérimentation était de comparer l'impact de patrons de conception sur un groupe d'experts ainsi que sur un groupe de novices. Lors de l'étude des menaces, je me suis rendu compte qu'en créant ces groupes moi-même, je pouvais introduire un grand nombre de biais. De même, demander aux sujets de se placer eux-mêmes dans un groupe peut introduire des biais.

J'ai dès lors décidé de prendre le problème dans l'autre sens. J'ai décidé de créer, à partir de parcours oculaires des sujets, deux groupes grâce à une technique de clustering et d'ensuite vérifier qu'il s'agisse d'un groupe d'experts et d'un groupe de novices.

L'algorithme utilisé pour créer les deux clusters est AGNES (Agglomerative Nesting). Il s'agit d'un algorithme hiérarchique qui utilise une matrice de dis-similarité.

[2] http ://www.pyxis-tech.com/fr/
[3] http ://www.eyelinkinfo.com/mount_main.php

Afin d'obtenir cette matrice de dis-similarité, j'ai d'abord tenté d'utiliser l'algorithme DTW (Dynamic Time Warping) qui est un algorithme de déformation temporelle dynamique. C'est-à-dire qu'il lui est possible de comparer la similitude de deux séquences indépendamment du temps. Il est par exemple utilisé dans la reconnaissance vocale ou d'écriture. Il permet dès lors d'obtenir une distance relative entre chaque paire de chemins oculaires. Malheureusement cet algorithme suppose une certaine similitude entre les séquences et la trop grande différence de durée entre les parcours oculaires rend impossible la détection de séquence (ou sous-séquence) utilisant cette technique. Sur un échantillon de 21 sujets, j'ai toujours obtenu un cluster de taille 1 et un cluster de taille 20.

Afin de pallier cette trop grande différence de durée d'examen des diagrammes entre les sujets, j'ai tenté de m'y prendre autrement. C'est la partie qui se déroule à Namur. Le but est toujours de comparer les chemins oculaires. L'idée est de comparer les polygones représentant l'enveloppe convexe des parties du diagramme les plus consultées par le sujet en question. Le diagramme est divisé en petites cellules. Pour chaque cellule, je compte le nombre de fixations présentes. Si la cellule contient moins de $x\%$ de l'ensemble des fixations, je ne considère pas cette cellule. Je calcule ensuite la plus petite enveloppe convexe contenant l'ensemble des cellules gardées comme intéressantes. Enfin, la distance de Hausdorff me permet de comparer les polygones deux à deux. Cette distance me permet de remplir la matrice de dis-similarité nécessaire à l'utilisation de l'algorithme AGNES et donc de construire mes 2 clusters en fonction de l'effort de recherche effectué par les sujets.

G.7 Conclusion

Par ce rapport, j'ai tenté de donner un aperçu réaliste du travail effectué durant mon stage à l'École Polytechnique de Montréal. Il y a évidemment bien encore des choses à dire concernant notamment le caractère humain d'une telle expérience. Celle-ci m'a enrichi à de multiples points de vue. J'en garderai un très bon souvenir.

J'ai tenté de décrire aussi fidèlement que possible les objectifs qui m'avaient été donnés, l'environnement dans lequel je me suis trouvé, la manière dont le travail s'est organisé, les outils utilisés ainsi que l'état d'avancement du travail lorsque j'ai quitté Montréal.

Il est évident qu'il ne s'agit ici que d'une description très brève du travail effectué. Pour plus d'information je vous renverrai donc vers mon futur mémoire.